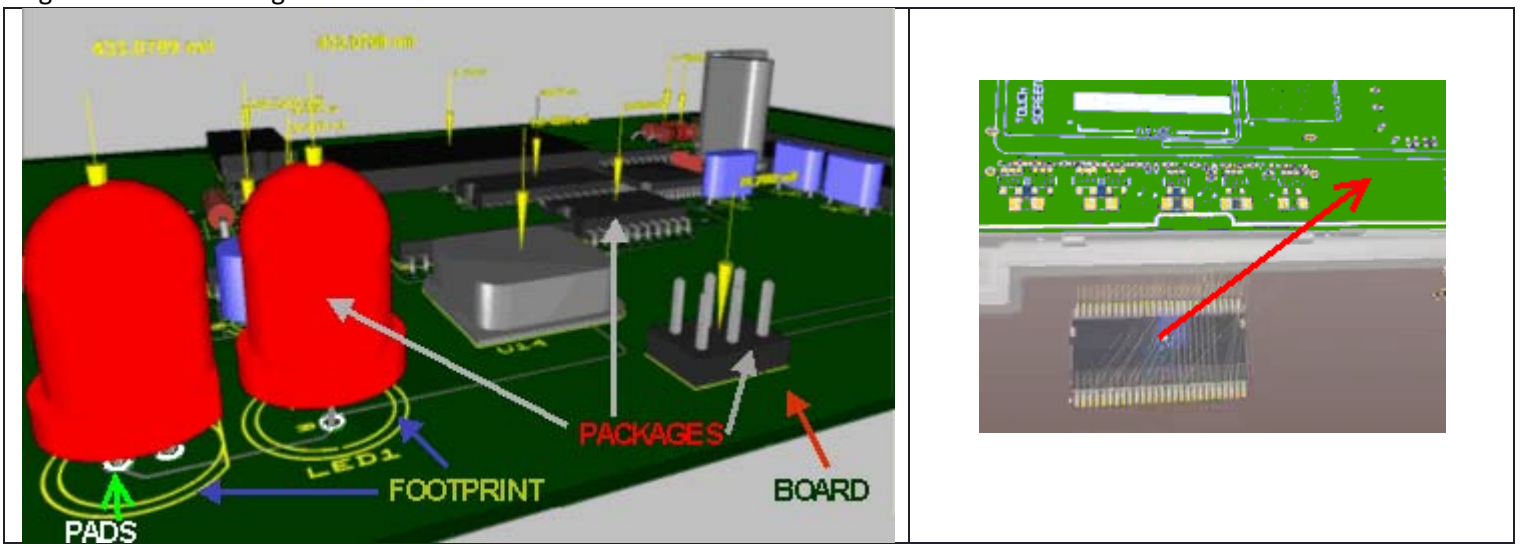
TUTORIAL PARA LA CREACIÓN DE COMPONENTES ELÉCTRICOS Y ELECTRÓNICOS EN 3D CON PovRay

Por Cesar David Restrepo Aristizabal cdavidrp@yahoo.com Tutorial versión PRELIMINAR 0.08 Abril 30 del 2009

El presente tutorial tiene como finalidad, dar una introducción en la creación de componentes o paquetes eléctricos y electrónicos en 3D, utilizando como software de renderizado PovRay [1]. Siguiendo unas pequeñas reglas de codificación serviría los modelos creados como bibliotecas adicionales para los software EAGLE3D [2] WINTYPON [3] y stripboards [5]

Como se sabe los aparatos electrónicos tienen una placa base o también llamada circuito o Board donde se montan los componentes o paquetes eléctricos y electrónicos, hay software que crean una PCB o dibujo del circuito de conexiones con sus PADs (que alojan los PINES de los componentes) en 2D, es decir con la silueta o huella (FOOTPRINT) del componente como dibujo guía para ubicarlo en el Board, aunque actualmente, hay unos cuantos software que trabajan con una imagen en 3D del componente para posicionarlo en el Board.

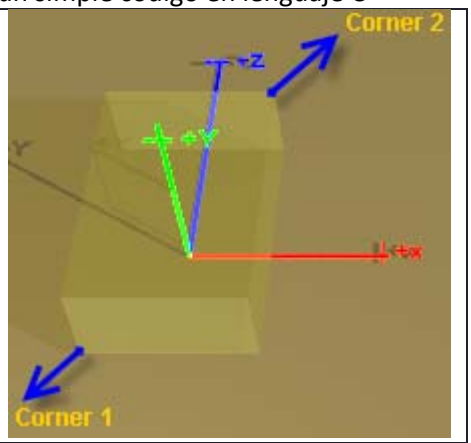
Figura #1 Altium Designer Software



El software EAGLE [4] y WINTYPON [3], crean una PCB en 2D solo con la silueta u huella (FOOTPRINT) del paquete, pero con el software gratuito EAGLE3D, permiten visualizar sus componentes en 3D utilizando el código generado por la PCB2D, unos macros de EAGLE3D y el Software de renderizado PovRay[1]

Figura #2: Un ejemplo de cómo se crea un caja en PovRay a partir de un simple código en lenguaje C

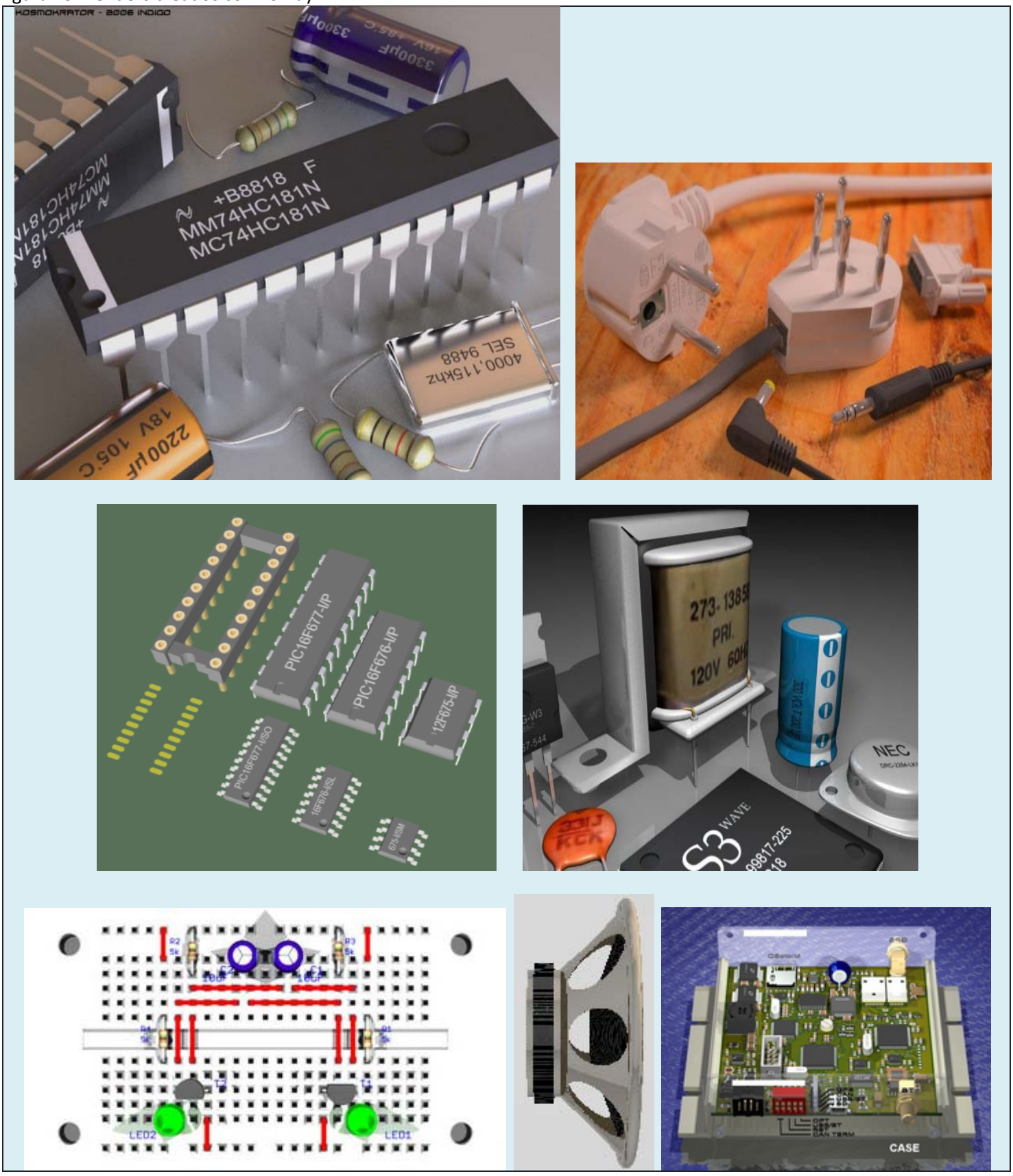
```
box {  // Inicio de la caja
        < Corner1 >,
        < Corner2 >
        [ Color Code ]
} // Fin de la caja

// con valores
box {  // Inicio de la caja
        <  -2.54, 0, -3.81  >,
        <  +2.54, 0, +3.81  >,
        pigment { color Transparent1 }
} // Fin de la caja
```



[1] www.povray.org/beta [2] www.matwei.de/doku.php?id=en:eagle3d:eagle3d
[3] www.typonrelais.com/wintypon_3d.htm [4] www.cadsoft.de [5] http://automagically.de/stripboard.html

Que se puede lograr codificando con PovRay y su imaginación

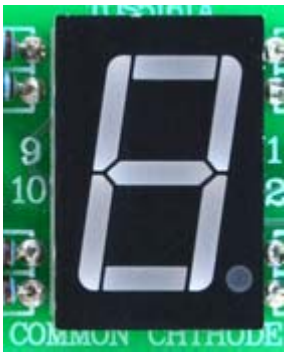
Figura #3: Renders creados con Povray



Espero que con estas imágenes los deje motivados a encaminarse con el Software PovRay

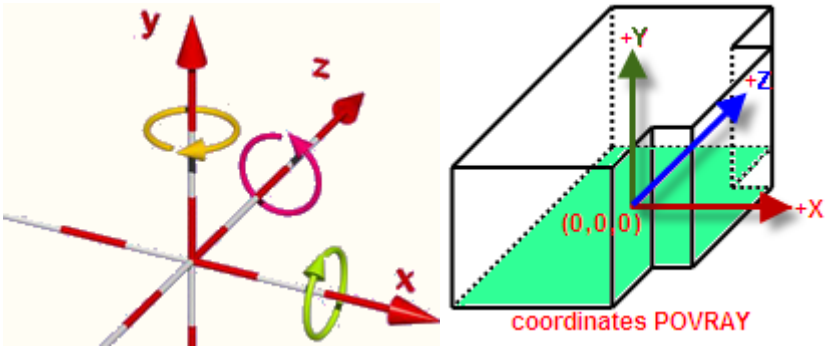
Nuestro primer componente a crear será un Display de 7 segmentos, dada su simplicidad geométrica. Una caja rectangular y una imagen plana en la parte superior.

Figura #4: Foto Real de un display de 7 segmentos common cathode



Se muestra a continuación el sistema coordenado de PovRay, donde el plano base es X_Z, aunque en matemáticas suele ser X_Y, de ahora en adelante se hablará de paquete en vez de componente o dispositivo eléctrico o electrónico.

Figura #5: Giro positivos y sistema coordenado de PovRay



Como el plano base es X_Z en PovRay, entonces se debe pensar la huella del paquete como X_Z y no como X_Y

Figura #6: Coordenadas de la huella de un paquete

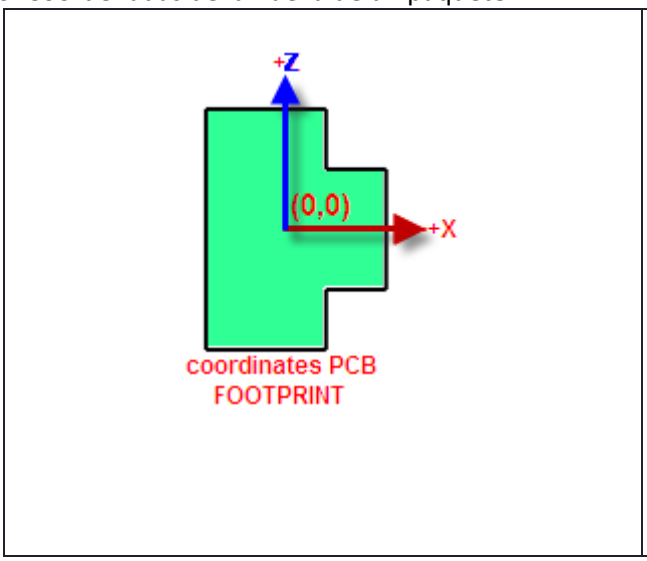
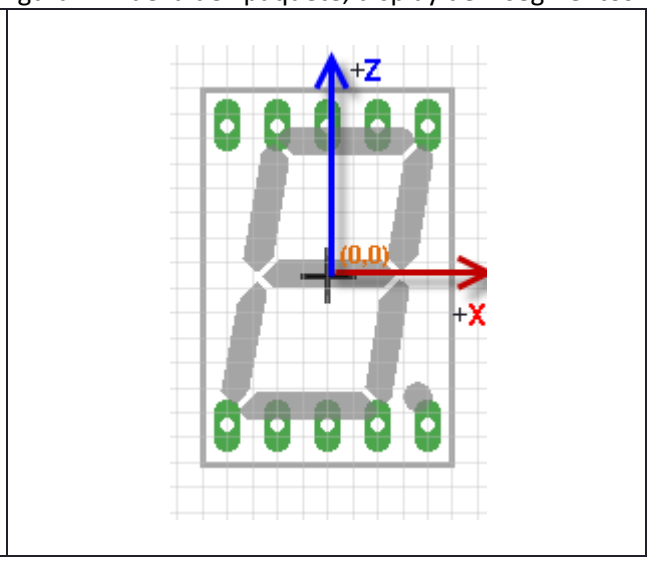


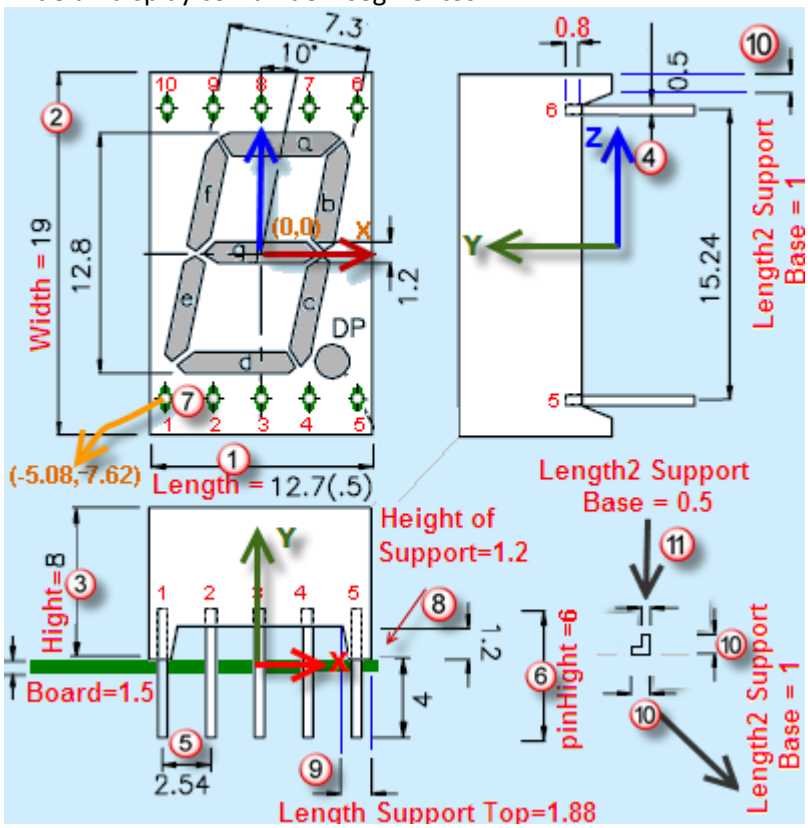
Figura #7 huella del paquete, display de 7 segmentos



En la Figura #7se aprecia la huella del Display de 7 segmentos que pone el software EAGLE, La cruz indica el origen de coordenadas de la huella, según esta trazamos los ejes coordenados X_Z para guiarnos en la construcción del paquete.

Según la hoja de datos que se puede encontrar en http://www.datasheetcatalog.com , un display de 7 segmentos tiene las siguientes dimensiones:

Figura #8: Dimensiones en mm de un display común de 7 segmentos



I: CODIFICACIÓN EN PovRay PARA CREAR EL PAQUETE MUESTRA, DISPLAY DE 7 SEGMENTOS

PASO#1, En un editor de texto o dentro de PovRay creamos un nuevo archivo en blanco para codificar nuestro Dispaly:

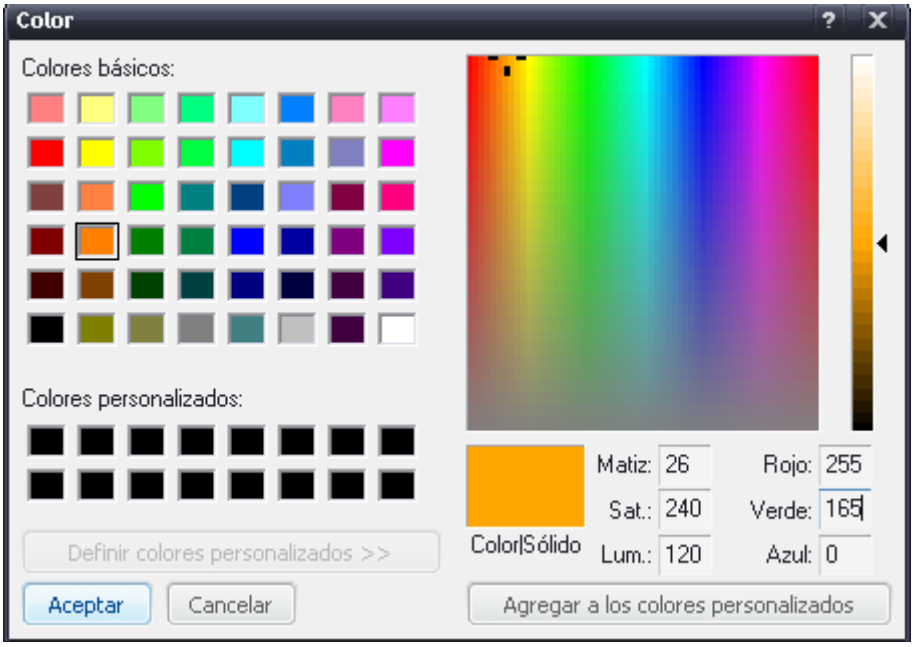
DEFINICIÓN DE LAS DIMENSIONES DEL PAQUETE EN MILÍMETROS (mm), La instrucción (#Local) define variables locales, que se utilizaran para la creación del paquete, hay que trabajar el código, mayormente posible con letras en vez de numérico, de esta manera se puede actualizar o hacer cambios más fáciles del código en un futuro.
Nota: (/*) y (* /) o (//) significan comentarios o guías del programador no ejecutadas por PovRay.

```
// Datasheet values
        // Box
#local L=12.7;              /* (1) Length_mm (Longitud del paquete) */
#local W=19;                /* (2) Width_mm  (Ancho del paquete) */
#local H=8;                 /* (3) Hight_mm  (Alto del paquete) */
        // Pin
#local pinDiameter = 0.5;       /* (4) Diameter of pins (Diámetro de los pines) */
#local pinSep = 2.54;           /* (5) Separation between pins (Separación entre pines) */
#local pinHight = 6;            /* (6) Total height of the pin (Altura total del pin)   */
#local pin1x = -5.08;           /* (7) X position of the first pin (Posición X del primer pin) */
#local pin1z = -7.62;           /* (7) Z position of the first pin (Posición Y del primer pin) */
        // Support
#local hSupport = 1.2;          /* (8) Height of Support (Altura del soporte) */
#local lSuppTop = 1.88;         /* (9) Length Support Top (longitud de la tapa del soporte) */
#local l1SuppBase = 1;          /* (10) Length1 Support Base (longitud1 de la base del soporte) */
#local l2SuppBase = 0.5;        /* (11) Length2 Support Base (longitud2 de la base del soporte) */
```

PASO#2

DEFINICIÓN DE COLORES, en un editor de gráficos se puede decodificar cualquier color en sus valores RED, GREEN, BLUE pero estos se deben dividir por 255 y aproximarlos mejor a dos cifras significativas, por ejemplo el color anaranjado seleccionado en la grafica siguiente, muestra RED=255, GREEN=165, BLUE=0 entonces el vector a definir tendrá los componentes con los siguientes valores RED=255/255=1, GREEN=165/255=0.6471~=0.65, BLUE=0/255=0, la sentencia final de declaración y asignación es: #local Orange01_rgb = rgb< 1, 0.65, 0 >;

Figura #9: Selector de colores



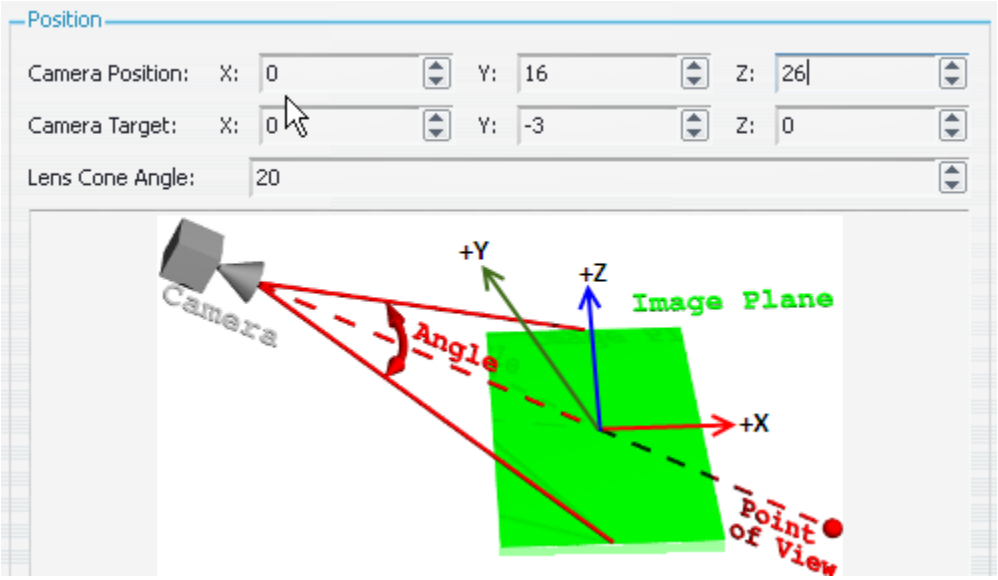

```
/* Definicion de colores */
        // RGB  Colors
#local Orange01_rgb      = rgb< 1, 0.65, 0 >;
#local Black_rgb         = rgb< 0, 0, 0 >;
#local Grey_rgb          = rgb< 0.5, 0.5, 0.5 >;
#local White_rgb         = rgb< 1, 1, 1 >;
#local Red_rgb           = rgb< 1, 0, 0 >;
#local Green0_5_rgb      = rgb< 0, 0.5, 0 >;
#local Blue_rgb          = rgb< 0, 0, 1 >;
#local Transparent1_rgbf  = rgbf< 0.90, 0.90, 0.90, 0.999 >; /* 4to valor es la cantidad de transparencia filtrada */
#local Col_Amber_88_rgbf  = rgbf< 0.90, 0.90, 0.50, 0.999 >;
```

Figura #10: Color y código de los colores básicos

PASO#3

AÑADIENDO LA CÁMARA, La estructura camera {} indica dónde está situada la cámara y cómo ésta ve la escena. La posición de la cámara se define con el siguiente vector posición < X_horizontal, Y_vertical, Z_profundidad >
donde -Z= La cámara se sale del plano X_Y, +Z= La cámara se introduce al plano X_Y, y se define con la sentencia location < X_horizontal, Y_vertical, Z_profundidad >, la sentencia look_at < X, Y, Z > indica hacia adonde apunta finalmente la escena.

Figura #11:



// Codigo

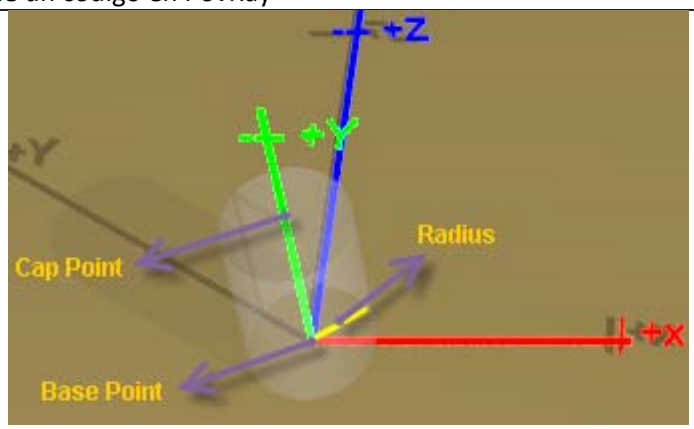
```
/* Suprimir para EAGLE3D */
////////////////////////////////
background { color White_rgb } // Fondo de la escena
camera {
        location< 0, 2*H, -2*W >       /* La cámara se ubica a dos veces la altura del paquete y acercada a dos veces el lado más largo del paquete*/
        look_at < 0, -(pinHight/2), 0 > /* punto de la escena a la que apunta finalmente. en este caso en el centro de los ejes coordenados pero desplazada   debajo del plano X_Z la mitad de la longitud del pin    */
}
light_source { <2, 4, -3> color White_rgb }
////////////////////////////////
```

PASO#4

DEFINICIÓN DE LOS OBJETOS QUE CONFORMAN EL PAQUETE DISPLAY DE 7 SEGMENTOS (Cilindros, Cajas, Soportes, etc.). El primer objeto a realizar del display será un pin de conexión eléctrica. Un pin es una figura geométrica, cilindro solido de color plata. El display de 7 segmentos posee 10 pines, todos del mismo tamaño y diámetro.

Figura #12: Un ejemplo de cómo se crea un cilindro a partir de un código en PovRay

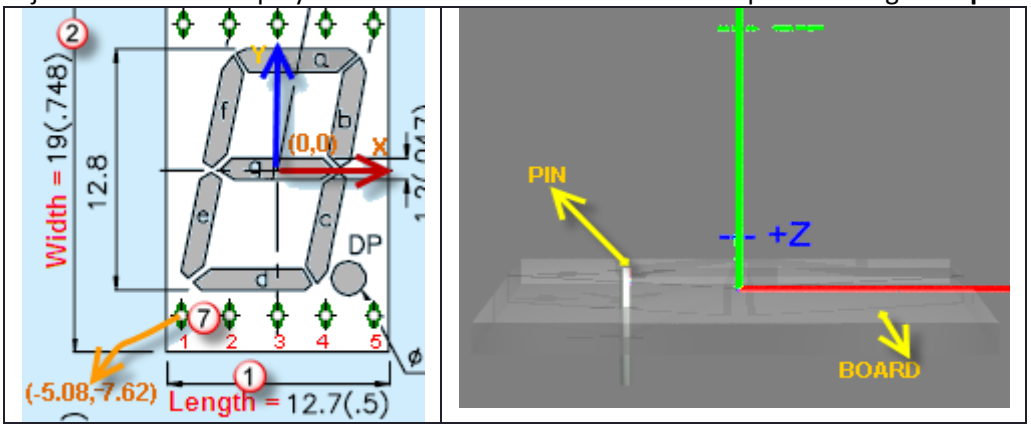
```
// Código

cylinder { // Inicio del cilindro
      < Base_Point >,
       < Cap_Point >,
       Radius
      [ open ]
      [ OBJECT_MODIFIERS... ]
} // Fin del cilindro
```



El pin con sus dimensiones se codifica una sola vez y se desplaza 10 veces para no repetir el mismo código cada vez que se lo requiera. El código en PovRay es el siguiente:

```
// Objects
/* Object #1: Código para crear un pin, cilindro de color plata de 6 mm de alto y 0.25mm de radio*/
#local onePin =                          // Nombre del objeto (un solo Pin)
        object {                         // Inicio del objeto (un solo Pin)
                cylinder {               // Inicio del cilindro
                        < 0, pinHight-4, 0 >,  /* Centro del disco de la tapa por encima (2mm) del plano X_Z o board */
                        < 0, pinHight-10, 0 >  /* Centro del  disco de la base, por debajo (-4mm) del plano X_Z */
                        pinDiameter/2          /* Radius of the cylinder (radio del cilindro) */
                        texture{ col_silver }  /* Color */
                }                        // Fin del cilindro
        }                                // Fin del objeto (un solo Pin)

// Código para ubicar un solo pin
object{ onePin translate < pin1x, 0,  pin1z > } /* (Pin base) Coloca el 1er pin trasladado hacia parte inferior izquierda del Display en la posición pin1x = -5.08, y=0, pin1z = -7.62 */
```

Figura #13: Primer objeto de nuestro Display un solo Pin eléctrico trasladado a su posición original < **pin1x, 0, pin1z** >



Código para poner el resto de pines, trasladando el objeto (onePin) a sus diferentes posiciones del Display

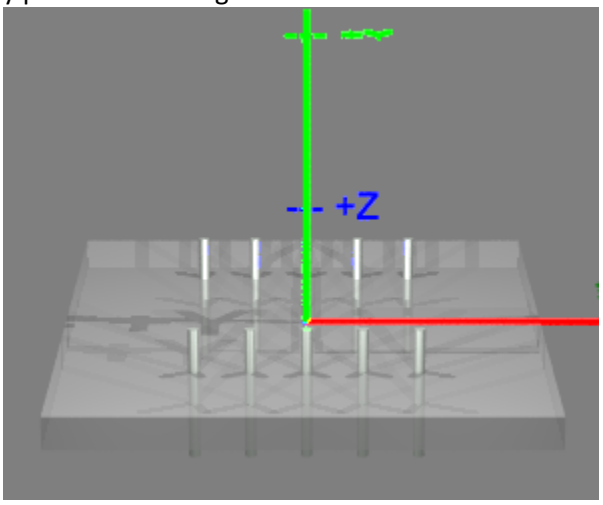
// Puesta de pines parte de abajo
object{ onePin translate < pin1x,0, pin1z> } /* (Pin base) Coloca el 1er pin parte inferior izquierda */
object{ onePin translate < pin1x+pinSep*1, 0, pin1z> } /* Coloca el 2do pin desplazandolo en X=2.54 mm a la derecha del Pin base */
object{ onePin translate < pin1x+pinSep*2, 0, pin1z> } /* Coloca el 3cer pin desplazandolo 5.08 mm (2.54*2) */
object{ onePin translate < pin1x+pinSep*3, 0, pin1z> } /* Coloca el 4to pin desplazandolo 7.62 mm (2.54*3) */
object{ onePin translate < pin1x+pinSep*4, 0, pin1z> } /* Coloca el 5to pin desplazandolo 10.16 mm (2.54*4) */

// Puesta de pines parte de arriba (por simetria)
object{ onePin translate < pin1x+pinSep*4, 0, -pin1z> } /* Coloca el 6to pin desplazandolo 10.16 mm parte superior derecha del Pin base y positiva la corrdenada Z */
object{ onePin translate < pin1x+pinSep*3, 0, -pin1z> } /* Coloca el 7mo pin desplazandolo 7.62 mm */
object{ onePin translate < pin1x+pinSep*2, 0, -pin1z> } /* Coloca el 8vo pin desplazandolo 5.08 mm */
object{ onePin translate < pin1x+pinSep*1, 0, -pin1z> } /* Coloca el 9no pin desplazandolo 2.54 mm */
object{ onePin translate < pin1x,0, -pin1z> } /* Coloca el 10mo pin */

Captura coloreada desde PovRay

```
// Pines parte de abajo
object{ onePin translate < pin1x,0, pin1z> }  /* (Pin base) Coloca el 1er pin parte inferior izquierda */
object{ onePin translate < pin1x+pinSep*1,  0, pin1z> }  /* Coloca el 2do pin desplazandolo en X=2.54 mm a la d
object{ onePin translate < pin1x+pinSep*2, 0, pin1z> }  /* Coloca el 3cer pin desplazandolo 5.08 mm (2.54*2) */
object{ onePin translate < pin1x+pinSep*3, 0, pin1z> }  /* Coloca el 4to pin desplazandolo 7.62 mm (2.54*3) */
object{ onePin translate < pin1x+pinSep*4, 0, pin1z> }  /* Coloca el 5to pin desplazandolo 10.16 mm (2.54*4) */

// Pines parte de arriba (por simetria)
object{ onePin translate < pin1x+pinSep*4, 0, -pin1z> }  /* Coloca el 6to pin desplazandolo 10.16 mm parte supe
object{ onePin translate < pin1x+pinSep*3, 0, -pin1z> }  /* Coloca el 7mo pin desplazandolo 7.62 mm  */
object{ onePin translate < pin1x+pinSep*2, 0, -pin1z> }  /* Coloca el 8vo pin desplazandolo 5.08 mm */
object{ onePin translate < pin1x+pinSep*1,  0, -pin1z> }  /* Coloca el 9no pin desplazandolo 2.54 mm */
object{ onePin translate < pin1x,0, -pin1z> }                        /* Coloca el 10mo pin */
```

Figura #14: Todos los pines del Display puestos en su lugar exacto



PASO#4.1

COLOCANDO LAS BASES DEL DISPLAY, Las bases del Display tienen forma de L, con el objeto PRISM permite a partir de una serie de puntos ubicados en el plano X_Z, crear un objeto solido de n lados y proyectados en el eje Y

```
/* Object #2:  código  para crear el soporte en L del display */
#local oneSupport =
object {
        prism {
                linear_sweep
                linear_spline
                hSupport,          /* altura por encima (2mm) del plano X_Z o board */
                0,                 /* altura por debajo (0mm) del plano X_Z o board */
                6,                 /* número de puntos de la huella en forma de L*/
                < 0, 0 >,              /* 1er coordenada del plano X_Z */
                < l1SuppBase, 0 >,     /* 2da coordenada del plano X_Z */
                < l1SuppBase, l2SuppBase >,  /* 3ra coordenada del plano X_Z */
                < l2SuppBase, l2SuppBase >,  /* 4ta coordenada del plano X_Z */
                < l2SuppBase, l1SuppBase >,  /* 5ta coordenada del plano X_Z */
                < 0, l1SuppBase >            /* 6ta coordenada del plano X_Z */
                pigment { OldGold }  /* Color */
        }
}
```

Figura #15: Segundo objeto de nuestro Display, un solo soporte en el origen del plano X_Z

```
// código: Objeto en el origen, sin translación ni rotación

object{ oneSupport rotate<0, 0, 0>   translate< 0, 0, 0 > }
```

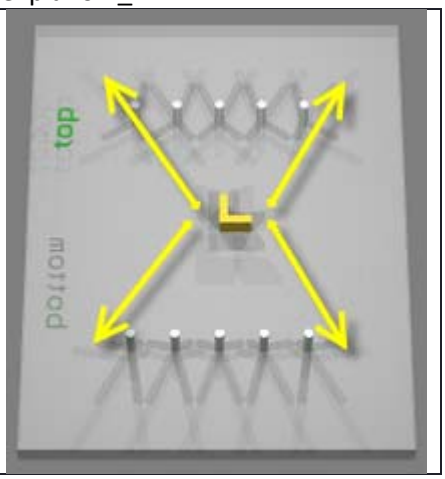
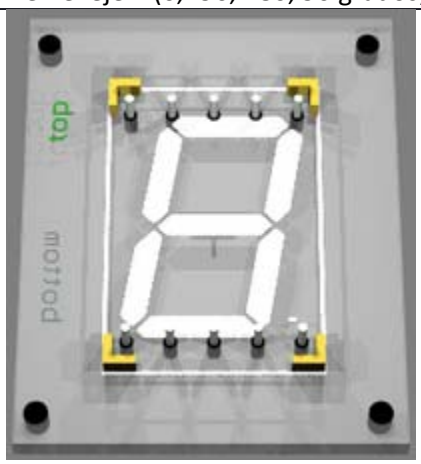


Figura #16: código soportes en su verdadera posición translación X_Z, y rotación en el eje Y (0, -90, 180, 90 grados)

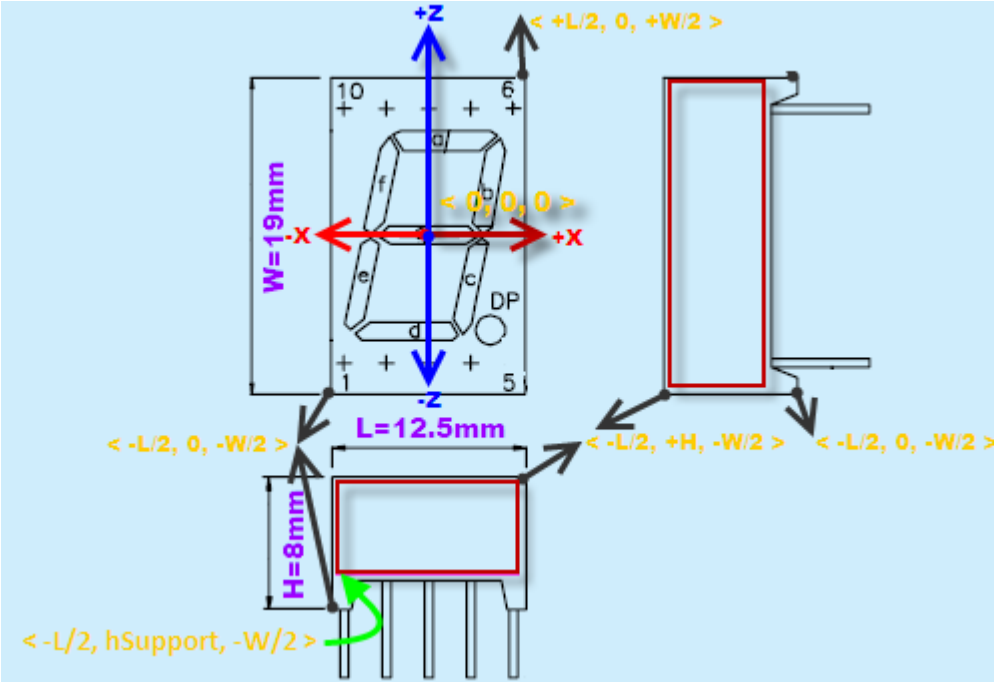
```
// Puesta de patillas de soporte

object{ oneSupport rotate<0, 0, 0>   translate< -L/2, 0, -W/2 > }
// esquina abajo izquierda

object{ oneSupport rotate<0, -90, 0> translate< L/2, 0, -W/2 > }
// esquina abajo derecha

object{ oneSupport rotate<0,180, 0>  translate< L/2, 0, W/2 > }
// esquina arriba izquierda

object{ oneSupport rotate<0, 90, 0>  translate< -L/2, 0, W/2 > }
// esquina arriba derecha
```



PASO#4.2

COLOCANDO LA CAJA DEL DISPLAY, hay que tener en cuenta que el centro de la base del display es el origen < 0, 0, 0 > de coordenadas, y que la caja se desplaza hacia arriba (**hSupport**) del plano X_Z debido a las patillas de soporte

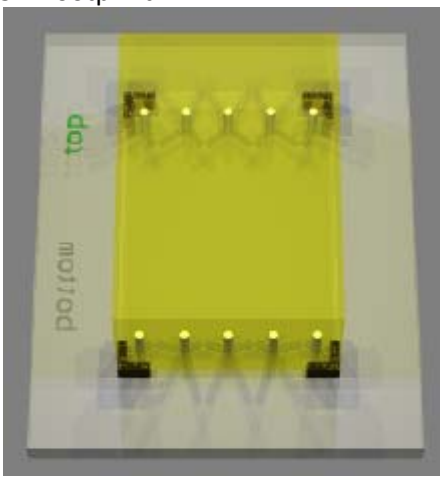
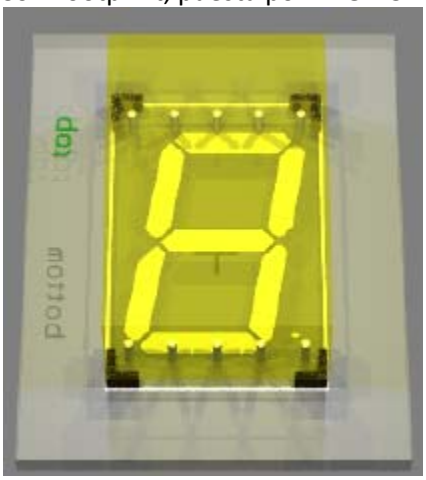
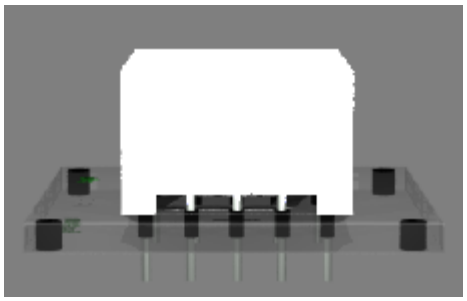
Figura #17: Coordenadas de la caja del display



A continuación se muestra el código de la caja del Display

```
/* Object #3: Caja01*/
#local box01 =
object{
        box {  // caja1 del componente
                < -L/2, hSupport, -W/2 >,          // Corner1
                < +L/2, +H, +W/2 >                 // Corner2
                pigment { Transparent1_rgbf }   // Color Code
        }     // Fin de la caja1
}

// Código para ubicar la caja
object{ box01 translate < 0, 0, 0> }
```

Figura #18: Imágenes de la caja

Sin Footprint	Con Footprint, puesta por EAGLE3D	Con los colores originales
		

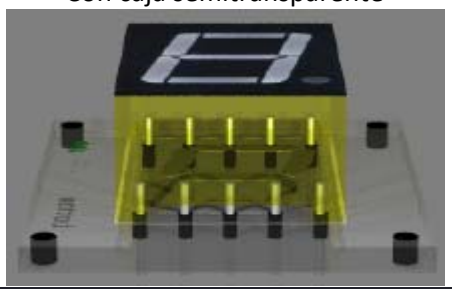
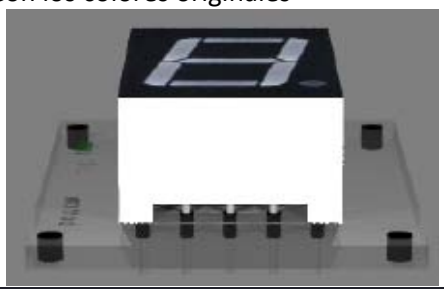
PASO#4.3

COLOCANDO LA PIEL (SKIN), hay una forma muy fácil de poner la tapa del paquete, simplemente llamado a un grafico en BMP, JPG o PNG y montándolo sobre la caja, la otra opción que es más elaborada, es crear con código la figura de 8 encima, en posteriores actualizaciones de este documento se hablara de aquello.

Para colocar una imagen como piel del paquete se debe crear una caja con una altura mínima de 0.1mm, la imagen por defecto se crea en el plano X_Y, por esta razón hay que rotarla 90 grados en +X, para que quede en el plano +X_+Z

```
/* Object #4: Caja02*/
#local box02 =
object{
        box { // Piel del paquete
                < -L/2, H, -W/2 >,
                < +L/2, H+0.1, +W/2 >
                pigment {
                        image_map{"skin7seg.jpg" once}
                        rotate< 90, 0, 0 > scale< 12.7, 1, 19 > translate< -L/2, 0, -W/2 >
                }
        }       // Fin de la caja2
}

// Código para ubicar la caja
object{ box02 translate < 0, 0, 0> }
```

Figura #19: Tapa o skin del paquete

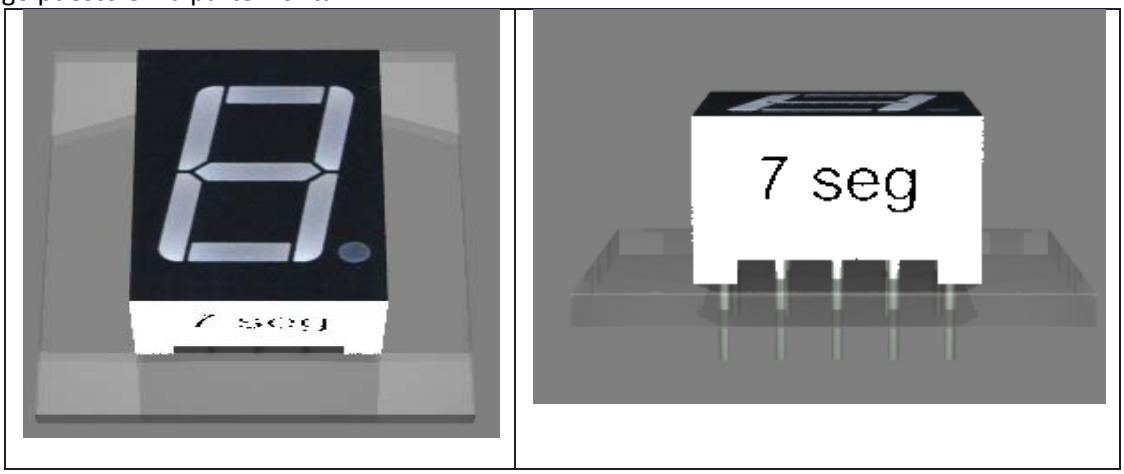
Con caja semitransparente	Con los colores originales
	

PASO#4.4

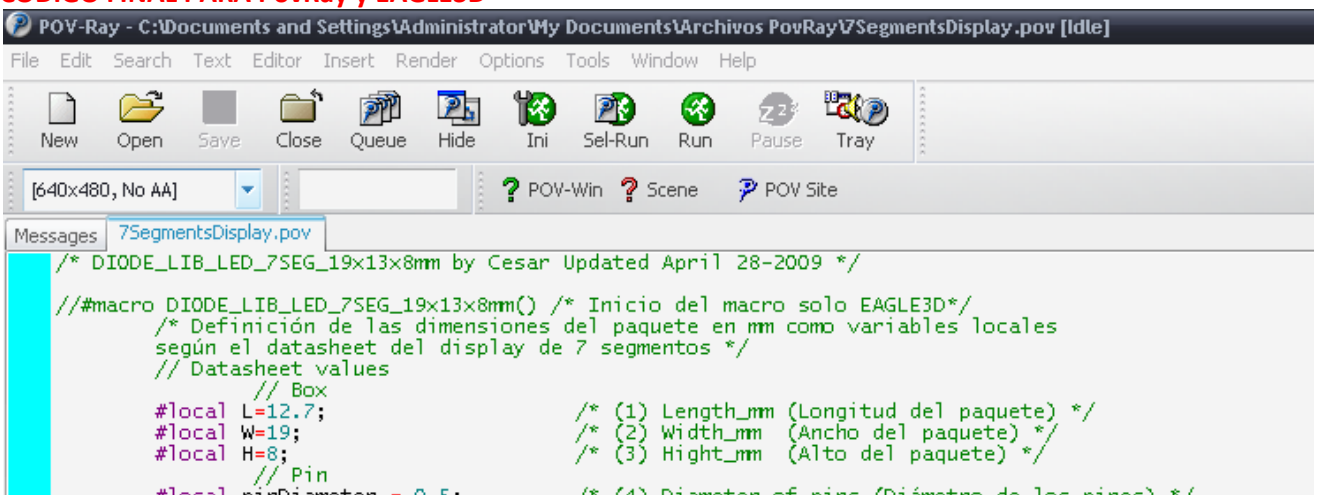
COLOCANDO UN TEXTO, PovRay coge el carácter de una fuente de texto y le crea una superficie o espesor, el texto ahora como imagen 3D, se ubica en el plano X_Y, por esta razón no requiere rotación, simplemente una translación a la esquina de la parte inferior izquierda.

```
/* Object5: código para poner un texto */
#local logo =
object{
        text {
                ttf             /* TrueType font */
                "arial.ttf",    /* font name */
                "  7 seg",      /* string name */
                0,              /* Espesor en Z, (The thickness of the extrusion in Z) */
                0               /* espacio entre caracteres */
                scale < 3, 3, 2 >
                pigment { Black }
        }
}

// Puesta del logo
object { logo rotate< 0, 0, 0 >  translate< -L/2+1, 4, -W/2 > } // esquina de la parte inferior izquierda
```

Figura #20: Logo puesto en la parte frontal



CODIGO FINAL PARA PovRay y EAGLE3D

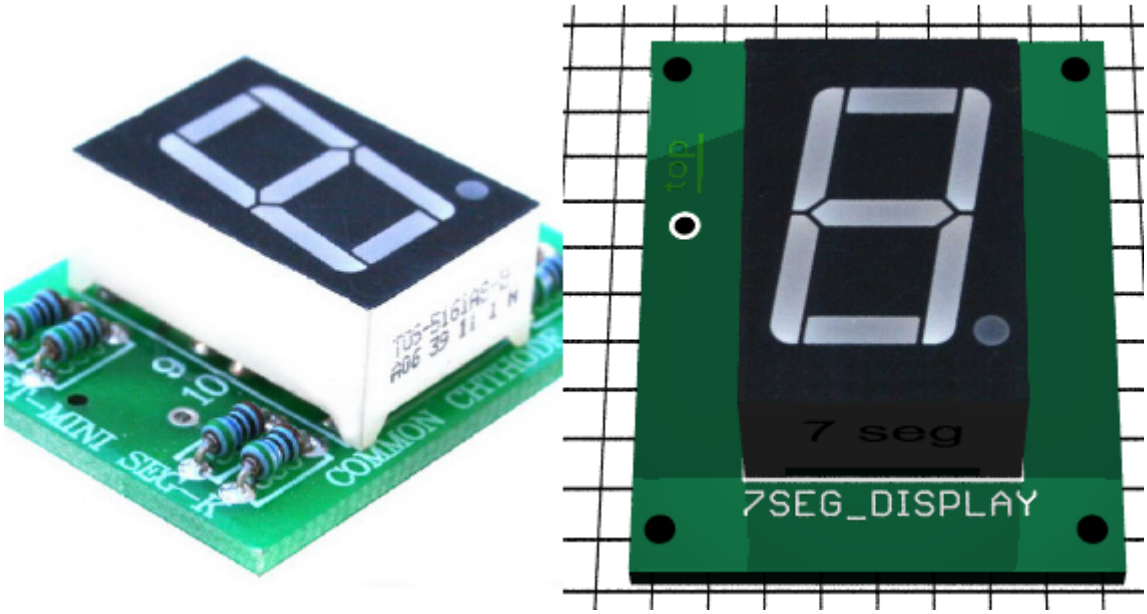


```
/* DIODE_LIB_LED_7SEG_19x13x8mm by Cesar Updated April 30-2009 */

//#macro DIODE_LIB_LED_7SEG_19x13x8mm() /* Inicio del macro solo EAGLE3D*/
/* Definición de las dimensiones del paquete en mm como variables locales
Según el datasheet del display de 7 segmentos */

// Datasheet values
// Box
#local L=12.7;    /*  (1) Length_mm (Longitud del paquete) */
#local W=19;     /*  (2) Width_mm  (Ancho del paquete) */
#local H=8;       /*  (3) Hight_mm  (Alto del paquete) */
// Pin
#local pinDiameter = 0.5; /* (4) Diameter of pins (Diámetro de los pines) */
#local pinSep = 2.54;        /* (5) Separation between pins (Separación entre pines) */
#local pinHight = 6;          /* (6) Total height of the pin (Altura total del pin)   */
#local pin1x = -5.08;        /*  (7) X position of the first pin (Posición X del primer pin) */
#local pin1z = -7.62;        /*  (7) Z position of the first pin (Posición Y del primer pin) */
// Support
#local hSupport = 1.2;    /* (8) Height of Support (Altura del soporte) */
#local lSuppTop = 1.88;  /* (9) Length Support Top (longitud de la tapa del soporte) */
#local l1SuppBase = 1;     /* (10) Length1 Support Base (longitud1 de la base del soporte) */
#local l2SuppBase = 0.5;     /* (10) Length2 Support Base (longitud2 de la base del soporte) */

/* Definicion de colores */
// RGB  Colors
#local Orange01_rgb = rgb< 1, 0.65, 0 >;
#local Black_rgb        = rgb< 0, 0, 0 >;
#local Grey_rgb         = rgb< 0.5, 0.5, 0.5 >;
#local White_rgb        = rgb< 1, 1, 1 >;
#local Red_rgb          = rgb< 1, 0, 0 >;
#local Green0_5_rgb = rgb< 0, 0.5, 0 >;
#local Blue_rgb         = rgb< 0, 0, 1 >;
#local Transparent1_rgbf  = rgbf<0.90, 0.90, 0.90, 0.999 >;  // 4to valor es la cantidad de transparencia filtrada
#local Col_Amber_88_rgbf = rgbf<0.90, 0.90, 0.50, 0.999>;

/*  ////////////////////////
 Suprimir para EAGLE3D */
background { color White_rgb } // Fondo de la escena
camera {
      location < 0, 2*H, -2*W >           /* La cámara se ubica a dos veces la altura del paquete y acercada a dos veces el lado
más largo del paquete*/
      look_at  < 0, -(pinHight/2),  0 >   /* punto de la escena a la que apunta finalmente. en este caso en el centro de los ejes
coordenados pero desplazada   debajo del plano X_Z la mitad de la longitud del pin    */
  }
light_source {  <2, 4, -3> color White_rgb }
////////////////////////////

// Objects
/* Object #1: Código para crear un pin, cilindro de color plata de 6 mm de alto y 0.25mm de radio*/
#local onePin =                                    // Nombre del objeto (un solo Pin)
          object {                                 // Inicio del objeto (un solo Pin)
                    cylinder {                     // Inicio del cilindro
```

```
                    < 0, pinHight-4, 0 >,   /* Centro del disco de la tapa por encima (2mm) del plano X_Z o board */
                    < 0, pinHight-10, 0 >   /* Centro del  disco de la base, por debajo (-4mm) del plano X_Z */
                    pinDiameter/2           /* Radius of the cylinder (radio del cilindro) */
                    pigment { Black_rgb }  /* Color */
                }                           // Fin del cilindro
        }                                   // Fin del objeto (un solo Pin)

/* Object #2:  código para crear el soporte en L del display */
#local oneSupport =
object {
        prism {
                linear_sweep
                linear_spline
                hSupport,           /* altura por encima (2mm) del plano X_Z o board */
                0,                  /* altura por debajo (0mm) del plano X_Z o board */
                6,                  /* número de puntos de la huella en forma de L*/
                < 0, 0 >,               /* 1er coordenada del plano X_Z */
                < l1SuppBase, 0 >,    /* 2da coordenada del plano X_Z */
                < l1SuppBase, l2SuppBase >,   /* 3ra coordenada del plano X_Z */
                < l2SuppBase, l2SuppBase >,   /* 4ta coordenada del plano X_Z */
                < l2SuppBase, l1SuppBase >,   /* 5ta coordenada del plano X_Z */
                < 0, l1SuppBase >           /* 6ta coordenada del plano X_Z */
                pigment { White_rgb }   /* Color */
        }
}


/* Object #3: Caja01*/
#local box01 =
object{
        box {  // caja1 del componente
                < -L/2, hSupport, -W/2 >,            // Corner1
                < +L/2, +H, +W/2 >                    // Corner2
                pigment { Transparent1_rgbf }   // Color Code
        }     // Fin de la caja1
}

/* Object #4: Caja02*/
#local box02 =
object{
        box { // Piel del paquete
                < -L/2, H, -W/2 >,
                < +L/2, H+0.1, +W/2 >
                pigment {
                        image_map{"skin7seg.jpg" once}
                        rotate< 90, 0, 0 > scale< 12.7, 1, 19 > translate< -L/2, 0, -W/2 >
                }
        }       // Fin de la caja2
}
```

```
/* Object5: código para poner un texto */
#local logo =
object{
	text {
		ttf		/* TrueType font */
		"arial.ttf",	/* font name */
		"  7 seg",	/* string name */
		0,		/* Espesor en Z, (The thickness of the extrusion in Z) */
		0		/* espacio entre caracteres */
		scale < 3, 3, 2 >
		pigment { Black_rgb }  /* Color */
	}
}

// Union de objetos para crear el paquete
//union { // Inicio de la union solo EAGLE3D

object{ box01 translate < 0, 0, 0> }
object{ box02 translate < 0, 0, 0> }

// Puesta de pines parte de abajo
object{ onePin translate < pin1x,0, pin1z> }  /* (Pin base) Coloca el 1er pin parte inferior izquierda */
object{ onePin translate < pin1x+pinSep*1,  0, pin1z> }  /* Coloca el 2do pin desplazandolo en X=2.54 mm a la derecha del
Pin base */
object{ onePin translate < pin1x+pinSep*2, 0, pin1z> }  /* Coloca el 3cer pin desplazandolo 5.08 mm (2.54*2) */
object{ onePin translate < pin1x+pinSep*3, 0, pin1z> }  /* Coloca el 4to pin desplazandolo 7.62 mm (2.54*3) */
object{ onePin translate < pin1x+pinSep*4, 0, pin1z> }  /* Coloca el 5to pin desplazandolo 10.16 mm (2.54*4) */

// Puesta de pines parte de arriba (por simetria)
object{ onePin translate < pin1x+pinSep*4, 0, -pin1z> }  /* Coloca el 6to pin desplazandolo 10.16 mm parte superior derecha
del Pin base y positiva la corrdenada Z */
object{ onePin translate < pin1x+pinSep*3, 0, -pin1z> }  /* Coloca el 7mo pin desplazandolo 7.62 mm  */
object{ onePin translate < pin1x+pinSep*2, 0, -pin1z> }  /* Coloca el 8vo pin desplazandolo 5.08 mm */
object{ onePin translate < pin1x+pinSep*1,  0, -pin1z> }  /* Coloca el 9no pin desplazandolo 2.54 mm */
object{ onePin translate < pin1x,0, -pin1z> }		/* Coloca el 10mo pin */

// Puesta de patillas de soporte
object{ oneSupport rotate<0, 0, 0>   translate< -L/2, 0, -W/2 > } // esquina abajo izquierda
object{ oneSupport rotate<0, -90, 0> translate< L/2, 0, -W/2 > } // esquina abajo derecha
object{ oneSupport rotate<0,180, 0>  translate< L/2, 0, W/2 > } // esquina arriba izquierda
object{ oneSupport rotate<0, 90, 0>  translate< -L/2, 0, W/2 > } // esquina arriba derecha
// Puesta del logo
object { logo rotate< 0, 0, 0 >  translate< -L/2+1, 4, -W/2 > } // esquina de la parte inferior izquierda
//} // Fin de la union solo EAGLE3D
//#end  // Fin del macro solo EAGLE3D
```

Figura #21 Comparación entre imagen real y renderizada



NOTAS SOBRE POVRAY Versión 3.7 Beta 32 (abril del 2009) http://www.povray.org/beta

La nueva versión de PovRay 3.7 Beta 32 con respecto a la versión 3.6 difiere en la ubicación del ejecutable y el archivo povray.ini

Para que EAGLE3D trabaje bien entonces se debe adherir en el archivo povray.ini las dos siguientes líneas

Library_Path="D:\PCB\EAGLE3D\povray\"
Library_Path="D:\PCB\EAGLE3D\ulp\"

Por ejemplo si EAGLE3D está instalado en D:\PCB\EAGLE3D\ entonces

La ruta del archivo povray.ini en la version de PovRay 3.6 está en:
C:\Program Files\POV-Ray for Windows v3.6\renderer\povray.ini
y el ejecutable en:
C:\Program Files\POV-Ray for Windows v3.6\bin\pvengine.exe

Mientras que la ruta del archivo povray.ini en la version de PovRay 3.7 está en:
C:\Documents and Settings\Administrator\My Documents\POV-Ray\v3.7\ini\povray.ini
y el ejecutable en:
C:\Documents and Settings\Administrator\Application Data\POV-Ray\v3.7\bin\pvengine-sse2.exe


Sugerencias, correcciones, aditivos al tutorial, ideas, mas ejemplos, componentes desarrollados por usted, etc., todo es bienvenido

Solicitud: Si usted es un programador de computadoras con interfaces graficas, tengo una idea la cual es crear un programa que permita crear componentes en 3D de manera de plantillas, es decir que el usuario escoja una figura geométrica, (caja, cilindro, trapecio, etc. y por medio de plantillas ya predefinidas cree una aproximación del componente 3D, el contacto aparece al principio de este tutorial
Gracias

Foro donde se habla un poco más sobre **LA CREACIÓN DE COMPONENTES ELÉCTRICOS Y ELECTRÓNICOS EN 3D**
http://www.todopic.com.ar/foros/index.php?topic=25385.0

Este tutorial se creó con apoyo de

- **POV-Ray en Castellano** http://www.ignorancia.org/es/index.php?page=Traduccion
- **POV Objets** http://perso.numericable.fr/pboucheny/eagle3d/povobjets.htm
- **OTRO FABULOSO TUTORIAL** www.f - lohmueller.de
- **Otro tutoriales sobre la CREACIÓN DE COMPONENTES 3D**

http://www.typonrelais.com/download/wt3d_creation_modele.zip

http://perso.numericable.fr/pboucheny/eagle3d/e_creatconv.htm

http://blog.everythingrobotics.com/tutorials/eagle3d-tutorials/using-google-sketchup-to-create-components/

Otros links
http://www.txemijendrix.com
http://www.geocities.com/txemijendrix/tutoriales/splinemacro/index.html

Sueño con realizar este paquete en los próximos días