2022-2023

Tutor: Víctor Alvarez Solano

Actividad voluntaria grupal modelado

Modelado y Visualización gráfica

Fecha: 3 de abril de 2023

Miembros:

- **Javier Fernández Castillo**
- **Andrés Ruíz Domínguez**
- **Manuel Otero Barbasán**

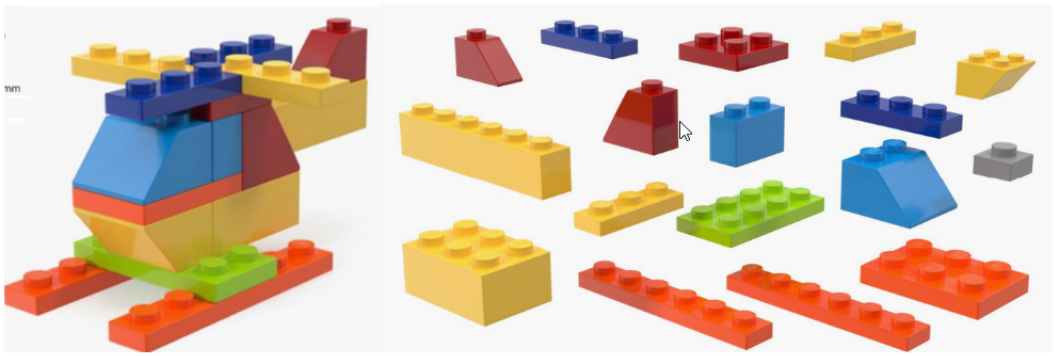
Índice

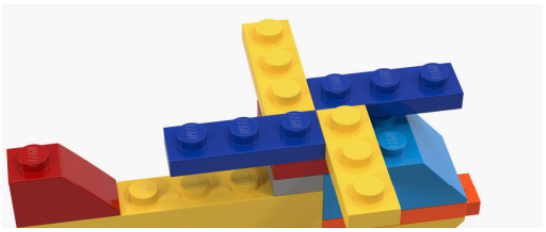
1.- Ejercicio 1: Figura de lego del helicóptero

Actividad Optativa

Problema 1

Construir un modelo en SAGE para el siguiente helicóptero, sabiendo que las dimensiones de la pieza base (1 cubo) son de 7.8 x 7.8 x 9.6 unidades, y que las piezas más planas tienen altura 1/3 de la usual (esto es, 3.2 unidades). NO SE CONSIDERAN los cilindros que permiten encajar las piezas entre sí; todas las piezas constan de 6 caras planas, para nuestros propósitos.



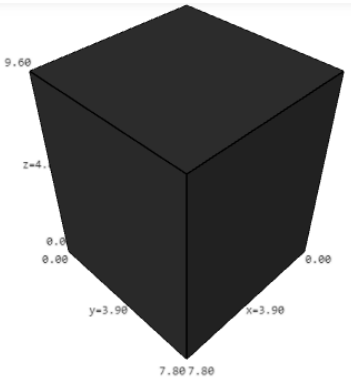


Variables globales y cargar las funciones que se necesitarán

In [1]:
```
#funciones de practicas
reset()
load('funciones.sage')

#transladar de medida de bloque en medida de bloque, solo tener cuidado con los tercios en z
def traslacion_pro(lista):
    return traslacion(matrix([lista[0]*width,lista[1]*length,lista[2]*height,1]).transpose())

#clase Poligono
from dataclasses import dataclass
@dataclass
class Poligono:
    vertices: matrix
    caras: matrix
#-----------------------------------------------------------------
#medidas de la base
width = 7.8;
length = 7.8;
height = 9.6;
tercio = 1/3;
```

Cubo base

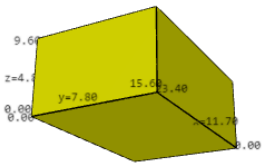
In [2]:
```
def get_vertices_cubo_base():
        coordenadas = [
        [0,0,0],[1,0,0],[1,1,0],[0,1,0],[0,0,1],[1,0,1],[1,1,1],[0,1,1]
        ]
        puntos_cubo_1 = matrix(coordenada+[1] for coordenada in coordenadas).transpose()
        return escalado([width,length,height])*puntos_cubo_1

def get_caras_cubo_base():
        caras = [
        [3, 2, 1, 0], [5, 4, 0, 1], [6, 7, 4, 5], [2, 3, 7, 6], [6, 5, 1, 2], [6, 5, 1, 2], [0, 4, 7, 3]
        ]
        return matrix(caras)

cubo_base = Poligono(vertices = get_vertices_cubo_base(),caras = get_caras_cubo_base())
dibujar_mallado_poligonal(cubo_base.vertices,cubo_base.caras,color="#333")
```



Piezas estandar

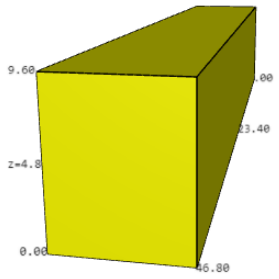
In [3]:
```
tres_x_dos= escalado([3,2,1])*cubo_base.vertices
dibujar_mallado_poligonal(tres_x_dos,cubo_base.caras,color="#dd0")
```

Out[3]:



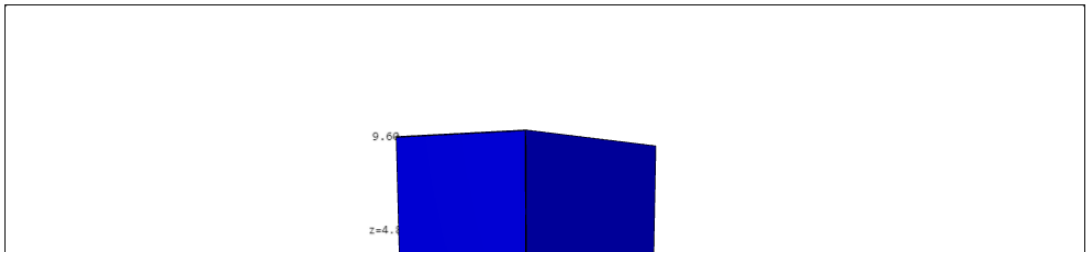
In [4]:
```
seis_x_uno= escalado([6,1,1])*cubo_base.vertices
dibujar_mallado_poligonal(seis_x_uno,cubo_base.caras,color="#dd0")
```

Out[4]:



In [5]:
```
dos_x_uno= escalado([2,1,1])*cubo_base.vertices
dibujar_mallado_poligonal(dos_x_uno,cubo_base.caras,color="#00d")
```

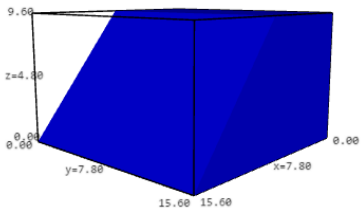
Out[5]:



Piezas diagonales

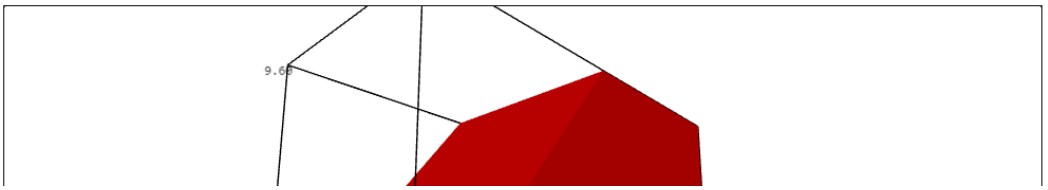
In [6]:
```
#reescala solo las coordenadas x de los puntos P5 y P6
P0_4 = dos_x_uno[:,0:5]
P5 = escalado([0.5,1,1])*dos_x_uno.column(5)
P6 = escalado([0.5,1,1])*dos_x_uno.column(6)
P7_8 = dos_x_uno[:,7:9]

diagonal_dos_x_dos = escalado([1,2,1])*P0_4.augment(P5).augment(P6).augment(P7_8)
dibujar_mallado_poligonal(diagonal_dos_x_dos,cubo_base.caras,color="#00d")
```

Out[6]:



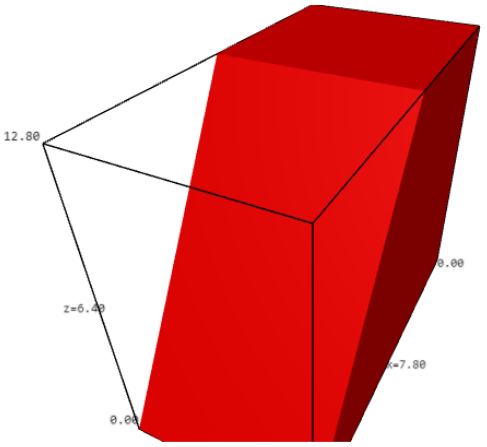
In [7]:
```
diagonal_dos_x_uno = escalado([1,0.5,1])*diagonal_dos_x_dos
dibujar_mallado_poligonal(diagonal_dos_x_uno,cubo_base.caras,color="#d00")
```

Out[7]:



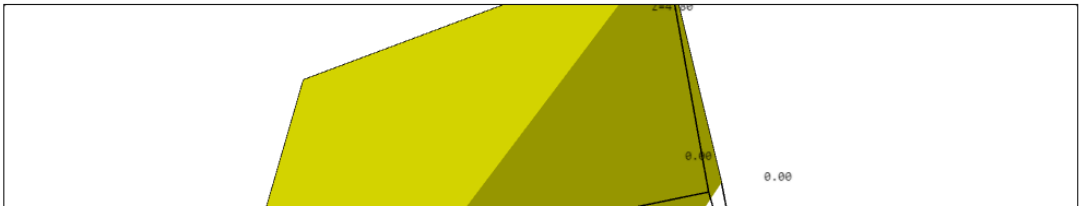
In [8]:
```
#Esta es un tercio mas alta de lo normal H=1+tercio
diagonal_dos_x_uno_y_tercio = escalado([1,0.5,1+tercio])*diagonal_dos_x_dos
dibujar_mallado_poligonal(diagonal_dos_x_uno_y_tercio,cubo_base.caras,color="#d00")
```

Out[8]:



In [9]:
```
N=matrix([0,0,1,0]).transpose() #primero simetria en z=0
D=matrix([0,0,height,0]).transpose() #para volver a los valores positivos de z
diagonal_inv_dos_x_dos = traslacion(D)*simetria(N)*diagonal_dos_x_dos
dibujar_mallado_poligonal(diagonal_inv_dos_x_dos,cubo_base.caras,color="#dd0")
```

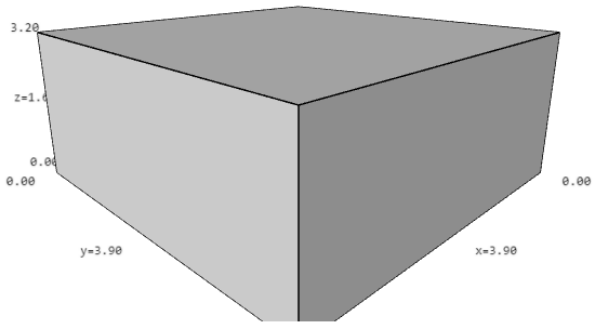
Out[9]:



Ahora las piezas de 1/3

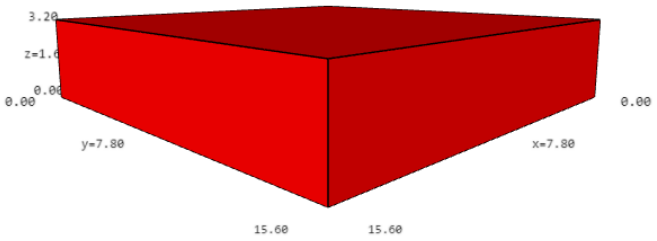
In [10]:
```
tercio_uno_x_uno= escalado([1,1,tercio])*cubo_base.vertices
dibujar_mallado_poligonal(tercio_uno_x_uno,cubo_base.caras,color="#ddd")
```

Out[10]:



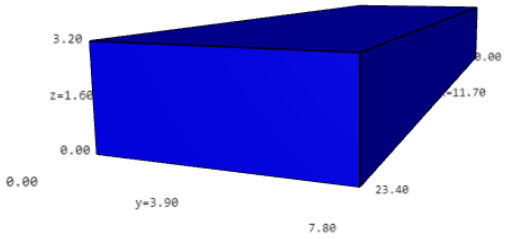
In [11]:
```
tercio_dos_x_dos= escalado([2,2,1])*tercio_uno_x_uno
dibujar_mallado_poligonal(tercio_dos_x_dos,cubo_base.caras,color="#f00")
```

Out[11]:



In [12]:
```
tercio_tres_x_uno= escalado([3,1,1])*tercio_uno_x_uno
dibujar_mallado_poligonal(tercio_tres_x_uno,cubo_base.caras,color="#00d")
```

Out[12]:

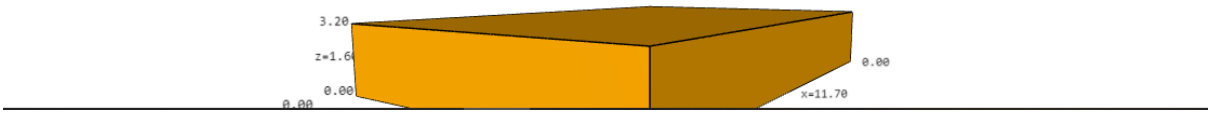


In [13]:
```
tercio_seis_x_uno= escalado([6,1,1])*tercio_uno_x_uno
dibujar_mallado_poligonal(tercio_seis_x_uno,cubo_base.caras,color="#fa0")
```

Out[13]:

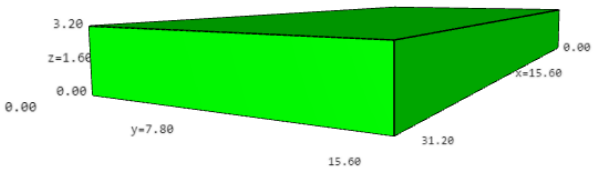
In [14]:
```
tercio_tres_x_dos= escalado([3,2,1])*tercio_uno_x_uno
dibujar_mallado_poligonal(tercio_tres_x_dos,cubo_base.caras,color="#fa0")
```

Out[14]:

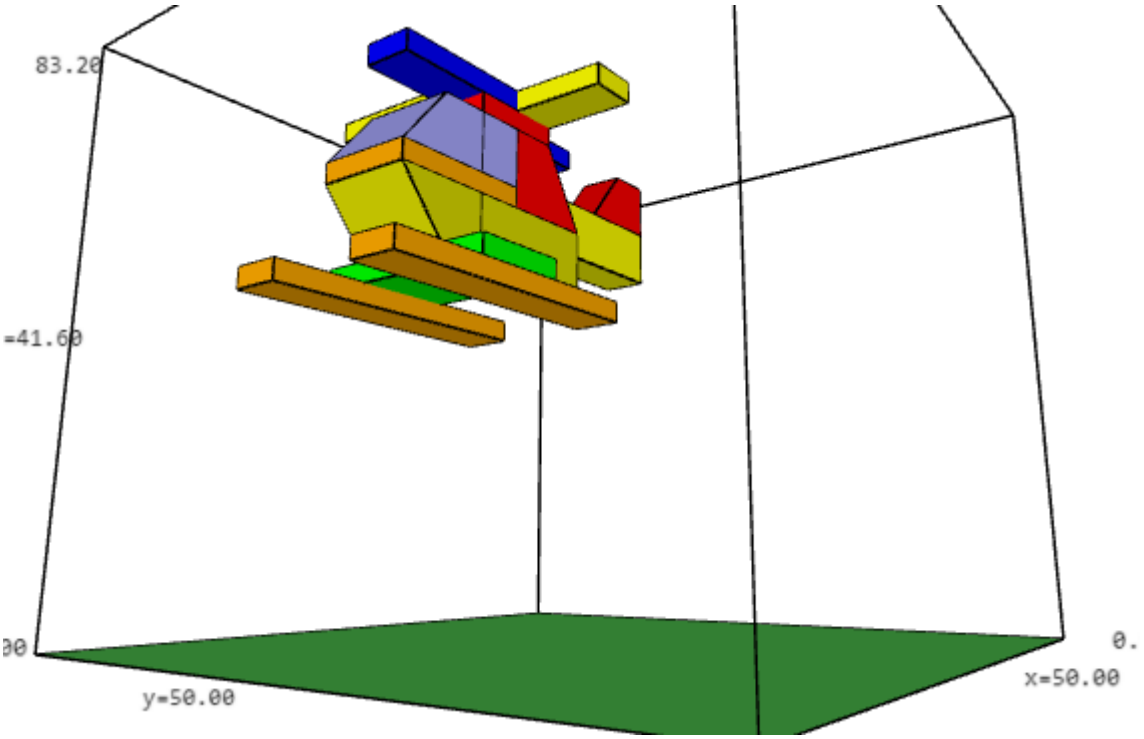


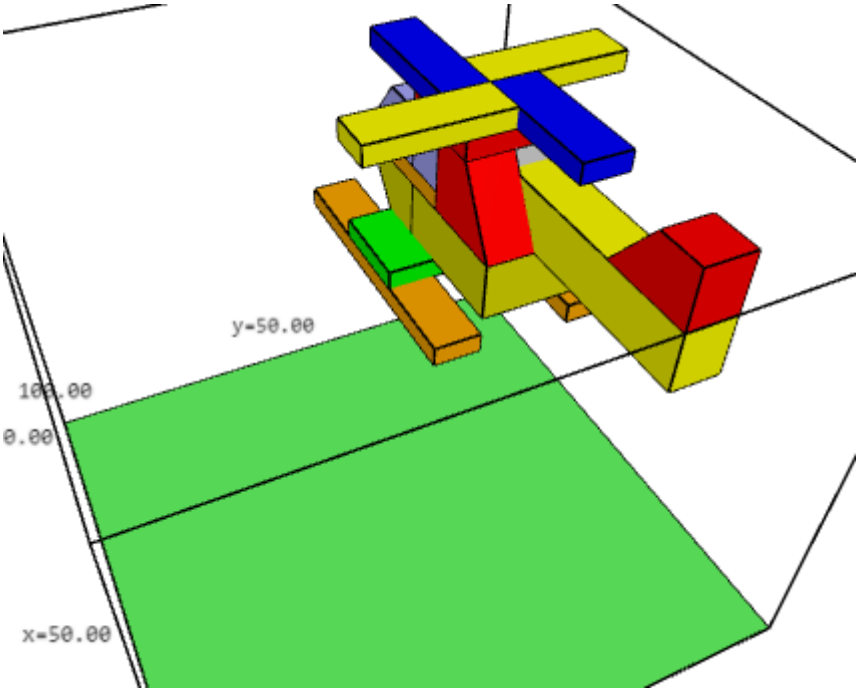
In [15]:
```
tercio_cuatro_x_dos= escalado([4,2,1])*tercio_uno_x_uno
dibujar_mallado_poligonal(tercio_cuatro_x_dos,cubo_base.caras,color="#0f0")
```

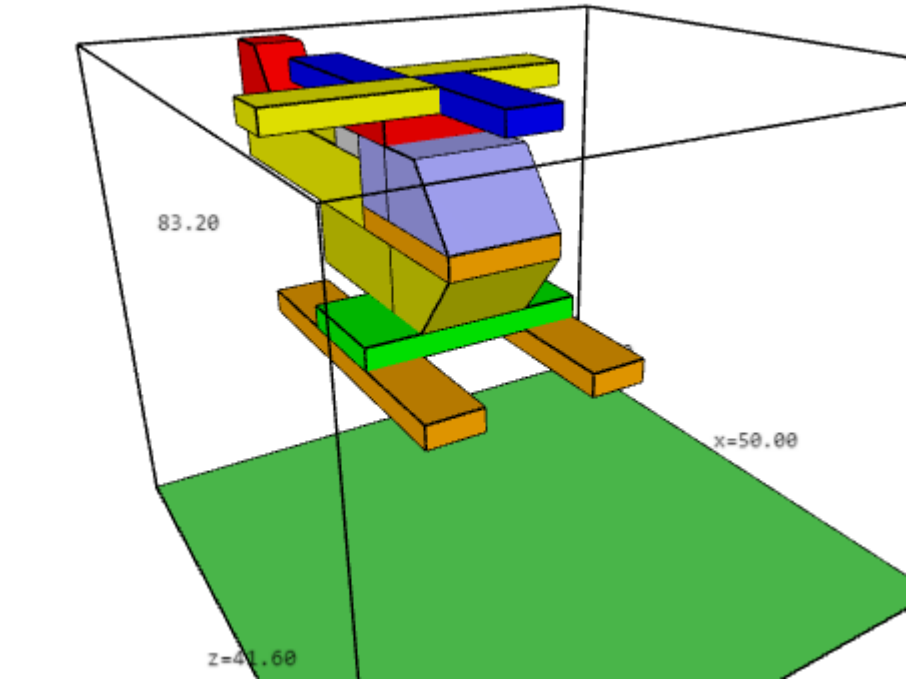
Out[15]:

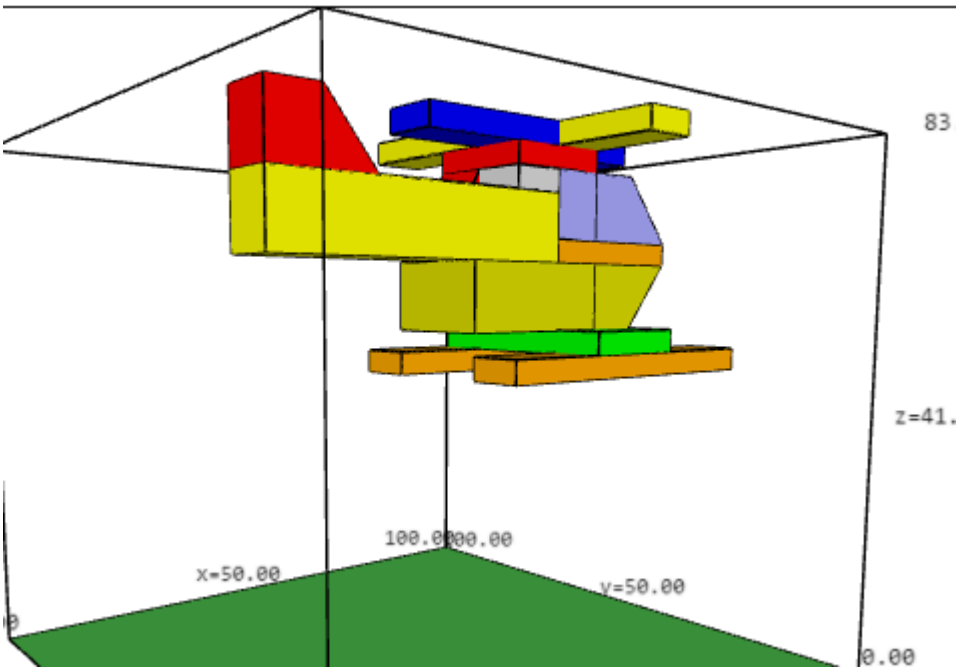


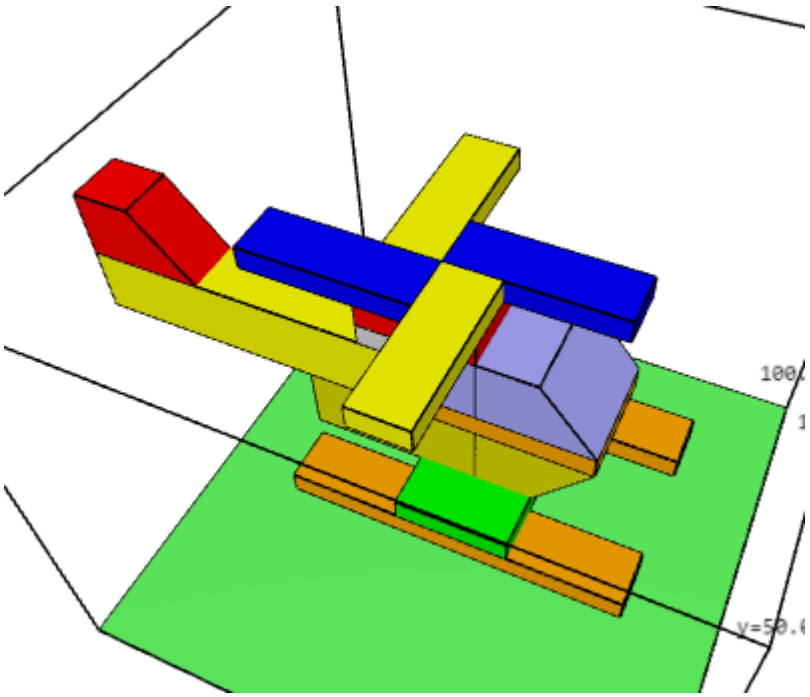
In [16]:
```
b=100
r = dibujar_mallado_poligonal_delimitado(matrix([[0,0,0,1],[b,0,0,1],[b,b,0,1],[0,b,0,1]]).transpose(),matrix([0,1,2,3]),color="
#es un helicoptero tiene que estar volando :P
r += dibujar_mallado_poligonal_delimitado(traslacion_pro([5,2,5,1])*tercio_seis_x_uno,cubo_base.caras,color="#fa0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([5,5,5,1])*tercio_seis_x_uno,cubo_base.caras,color="#fa0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([7,6,5+1*tercio,1])*rotacion('z',-pi/2)*tercio_cuatro_x_dos,cubo_base.ca
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([5,3,5+2*tercio,1])*tres_x_dos,cubo_base.caras,color="#dd0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([8,3,5+2*tercio,1])*diagonal_inv_dos_x_dos,cubo_base.caras,color="#dd0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([7,3,6+2*tercio,1])*tercio_tres_x_dos,cubo_base.caras,color="#fa0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([7,5,6+2*tercio,1])*rotacion('z',pi)*diagonal_dos_x_uno_y_tercio,cubo_ba
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([8,3,7,1])*rotacion('z',pi/2)*dos_x_uno,cubo_base.caras,color="#aaf")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([8,3,7,1])*diagonal_dos_x_dos,cubo_base.caras,color="#aaf")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([1,3,7-tercio,1])*seis_x_uno,cubo_base.caras,color="#ff0")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([6,3,8-tercio,1])*tercio_uno_x_uno,cubo_base.caras,color="#ddd")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([1,3,8-tercio,1])*diagonal_dos_x_uno,cubo_base.caras,color="#f00")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([6,3,8,1])*tercio_dos_x_dos,cubo_base.caras,color="#f00")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([4,3,8+tercio,1])*tercio_tres_x_uno,cubo_base.caras,color="#00f")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([7,4,8+tercio,1])*tercio_tres_x_uno,cubo_base.caras,color="#00f")
r+= dibujar_mallado_poligonal_delimitado(traslacion_pro([7,4,8+tercio,1])*rotacion('z',pi/2)*tercio_tres_x_uno,cubo_base.caras,c
```

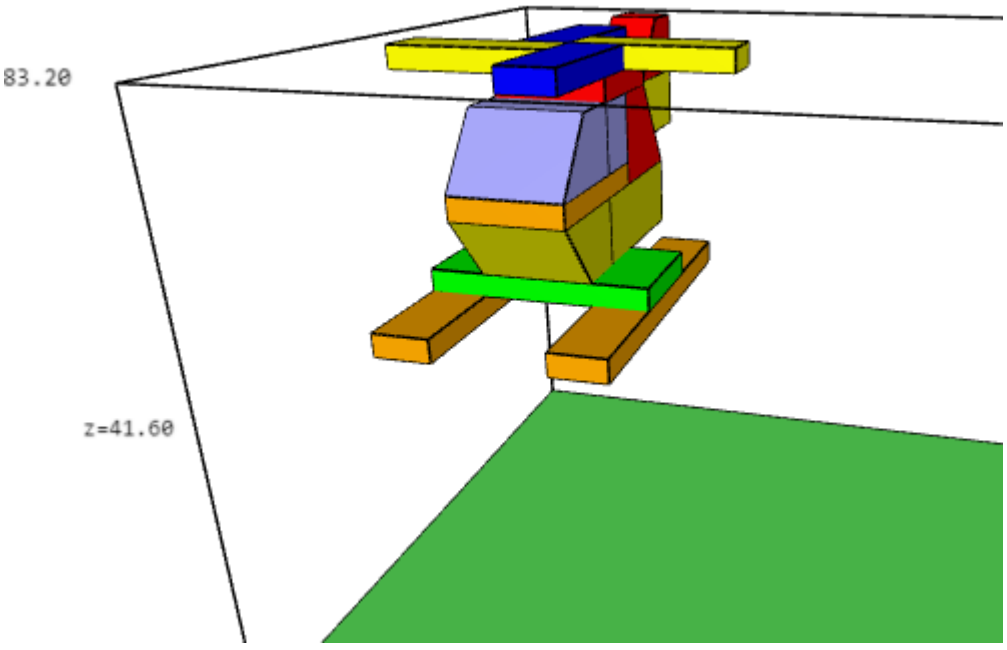

83.20
=41.60
00
y=50.00
0.
x=50.00


y=50.00
100.00
0.00
x=50.00


83.20
x=50.00
z=41.60


83
z=41.
100.0000.00
x=50.00
y=50.00
0.00




83.20
z=41.60

2.- Ejercicio 2: Figura de la mariposa

Problema 2

Construir un modelo 3D en SAGE para la mariposa

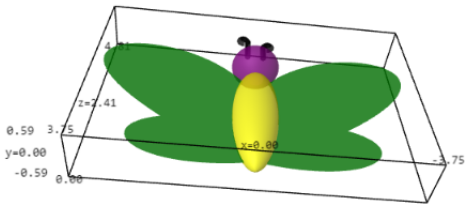
In [1]:
```
#funciones de practicas
reset()
load('funciones.sage')
```

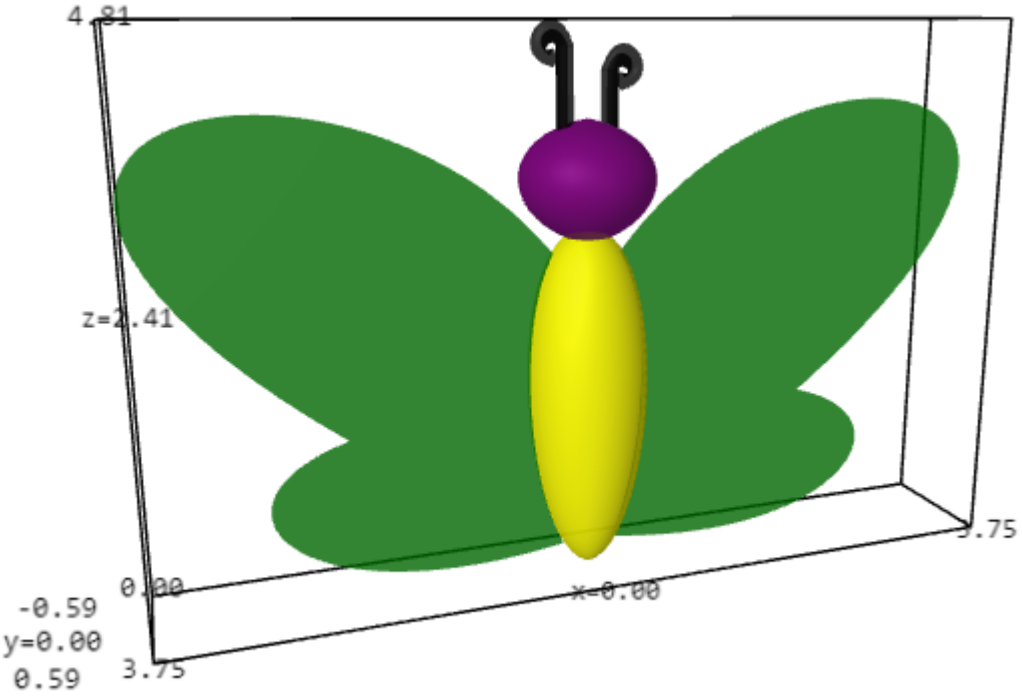
In [2]:
```
#EJERCICIO MARIPOSA
u,v=var('u,v')
#CUERPO
#2 curvas a trozos, 1 para el tronco, y otro para formar la cabeza
PC1=matrix(QQ,[[0,0,0,1],[0.55,0,0.15,1],[0.75,0,2.5,1],[0.2,0,3,1]]).transpose()
PC2=matrix(QQ,[[0.2,0,3,1],[0.75,0,3.25,1],[0.75,0,3.85,1],[0,0,4,1]]).transpose()

C1=lambda u: punto_curva_bezier(PC1,u)
C2=lambda u: punto_curva_bezier(PC2,u)

S1=superficie_revolucion_eje_z(C1)
S2=superficie_revolucion_eje_z(C2)

#ALA DEL EJE X POSITIVO
#4 curvas a trozos, superfice reglada entre la curva exterior superior con la curva superior interior.
#Superfice reglada entre la curva exterior inferior con la curva inferior interior
PG1=matrix(QQ,[[0.23,0,0.2,1],[0.38,0,0.5,1],[0.5,0,1.15,1],[0.5,0,1.42,1]]).transpose()
PG2=matrix(QQ,[[0.5,0,1.42,1],[0.55,0,2,1],[0.5,0,2.5,1],[0.3,0,2.85,1]]).transpose()
PC3=matrix(QQ,[[0.23,0,0.2,1],[2.6,0,-0.2,1],[3.5,0,1,1],[2.1,0,1.4,1]]).transpose()
PC4=matrix(QQ,[[2.1,0,1.4,1],[6,0,4.3,1],[2,0,5,1],[0.3,0,2.85,1]]).transpose()

curva1_exterior_ala1= lambda u: punto_curva_bezier(PC3,u)
curva2_exterior_ala1= lambda u: punto_curva_bezier(PC4,u)
curva1_interior_ala1 = lambda u: punto_curva_bezier(PG1,u)
curva2_interior_ala1 = lambda u: punto_curva_bezier(PG2,u)

S3=superficie_reglada(curva1_exterior_ala1,curva1_interior_ala1)
S4=superficie_reglada(curva2_exterior_ala1,curva2_interior_ala1)

#ALA DEL EJE X NEGATIVO
#Calculamos las 4 curvas necesarias para este ala haciendo una simetría de las
#4 curvas del lado positivo con respecto del plano x=0
r=matrix(QQ,[[1,0,0,0]]).transpose()

PG3= simetria(r)*PG1
PG4= simetria(r)*PG2
PC5= simetria(r)*PC3
PC6= simetria(r)*PC4

curva1_exterior_ala2= lambda u: punto_curva_bezier(PC5,u)
curva2_exterior_ala2= lambda u: punto_curva_bezier(PC6,u)
curva1_interior_ala2 = lambda u: punto_curva_bezier(PG3,u)
curva2_interior_ala2 = lambda u: punto_curva_bezier(PG4,u)

S5=superficie_reglada(curva1_exterior_ala2,curva1_interior_ala2)
S6=superficie_reglada(curva2_exterior_ala2,curva2_interior_ala2)


#ANTENA LARGA
#Usamos 2 superficies de bezier para el tronco de la antena y otras 2 para la curva
PB1=matrix([[0.2, 0.3 , 0.3 , 0.2, 0.2, 0.3 , 0.3 , 0.2, 0.2, 0.3 , 0.3 , 0.2 ,0.2, 0.3 , 0.3 , 0.2],
[-0.075 ,-0.05 , 0.05 , 0.075 , -0.075 ,-0.05 , 0.05 , 0.075, -0.075 ,-0.05 , 0.05 , 0.075 , -0.075 ,-0.05 , 0.05 , 0.075],
[3.9 , 3.9 , 3.9 , 3.9 , 4.2 , 4.2 , 4.2 , 4.2 , 4.4 ,4.4 , 4.4 , 4.4 , 4.6 , 4.6 , 4.6 , 4.6],
[1  ,1  ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ]])

SB1=superficie_bezier(PB1)

PB2=matrix([[0.2, 0.1 , 0.1 , 0.2, 0.2, 0.1 , 0.1 , 0.2, 0.2, 0.1 , 0.1 , 0.2 ,0.2, 0.1 , 0.1 , 0.2],
[0.075 ,0.05 , -0.05 , -0.075 , 0.075 ,0.05 , -0.05 , -0.075, 0.075 ,0.05 , -0.05 , -0.075 , 0.075 ,0.05 , -0.05 , -0.075],
[3.9 , 3.9 , 3.9 , 3.9 , 4.2 , 4.2 , 4.2 , 4.2 , 4.4 ,4.4 , 4.4 , 4.4 , 4.6 , 4.6 , 4.6 , 4.6],
[1  ,1  ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ]])

SB2=superficie_bezier(PB2)

PB5=matrix([
[0.2, 0.1 , 0.1 , 0.2, 0.3, 0.3 ,0.3 , 0.3 , 0.6 , 0.8, 0.8 , 0.6 , 0.3 , 0.3 , 0.3 , 0.3],
[0.075 ,0.05 , -0.05 , -0.075,0.075 ,0.05 , -0.05 , -0.075,0.075 ,0.05 , -0.05 , -0.075,0.075 ,0.05 , -0.05 , -0.075],
[4.6 , 4.6 , 4.6 , 4.6 , 4.9 , 5.2 , 5.2 , 4.9 , 4.6 , 4.5 , 4.5 , 4.6 , 4.5 , 4.4 , 4.4, 4.5 ],
[1  ,1  ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ]])

SB5=superficie_bezier(PB5)

PB6=matrix([
[0.2, 0.3 , 0.3 , 0.2, 0.3, 0.3 ,0.3 , 0.3 , 0.6 , 0.4, 0.4 , 0.6 , 0.3 , 0.3 , 0.3 , 0.3],
[-0.075 ,-0.05 , 0.05 , 0.075,-0.075 ,-0.05 , 0.05 , 0.075,-0.075 ,-0.05 , 0.05 , 0.075,-0.075 ,-0.05 , 0.05 , 0.075],
[4.6 , 4.6 , 4.6 , 4.6 , 4.9 , 4.6 , 4.6 , 4.9 , 4.6 , 4.5 , 4.5 , 4.6 , 4.5 , 4.6 , 4.6, 4.5 ],
[1  ,1  ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ,1 ,1  ,1  ,1  ,1  ]])

SB6=superficie_bezier(PB6)
```

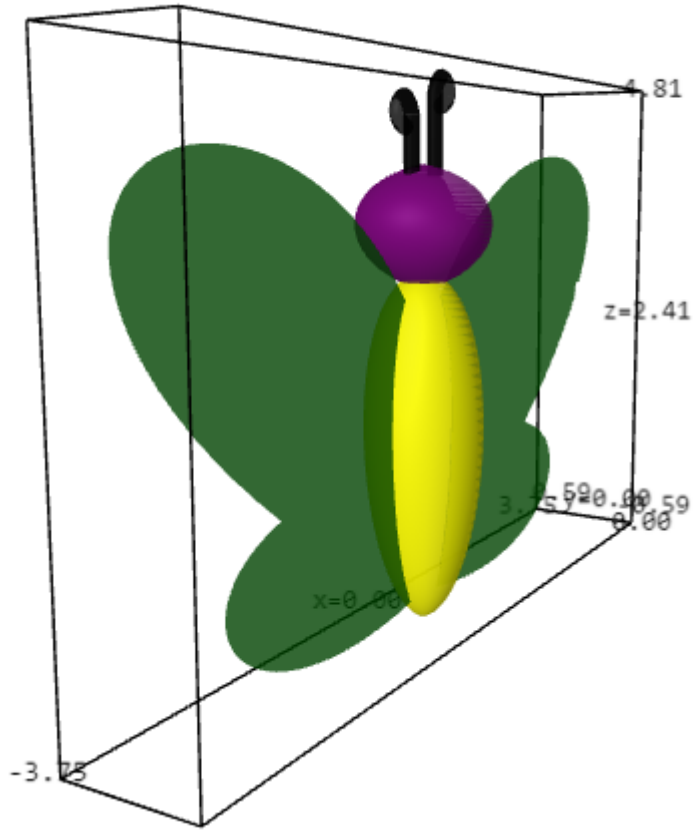
```
#ANTENA CORTA -> usamos de nuevo la simetria
Auxiliar= matrix([[0,  0, 0 , 0, 0, 0 , 0, 0, 0 , 0, 0 , 0 ,0, 0 , 0 , 0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0.2  , 0.2 ,0.2  ,0.2  ,0.2  ,0.2  ,0.2 ,0.2 ,0.2  ,0.2  ,0.2  ,0.2  ,0.2 ,0.2  ,0.2 ,0.2  ,0.2  ,0.2  ],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ])

Auxiliar2= matrix([[0,  0, 0 , 0, 0, 0 , 0, 0, 0 , 0, 0 , 0 ,0, 0 , 0 , 0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0  , 0 ,0  ,0  ,0.2  ,0.2 ,0.2 ,0.2  ,0.2  ,0.2  ,0.2  ,0.2 ,0.2  ,0.2 ,0.2  ,0.2  ,0.2  ],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ])

PB3=(simetria(r)*PB1)-Auxiliar2
SB3=superficie_bezier(PB3)

PB4=(simetria(r)*PB2)-Auxiliar2
SB4=superficie_bezier(PB4)

PB7=(simetria(r)*PB5)-Auxiliar
SB7=superficie_bezier(PB7)

PB8=(simetria(r)*PB6)-Auxiliar
SB8=superficie_bezier(PB8)
```

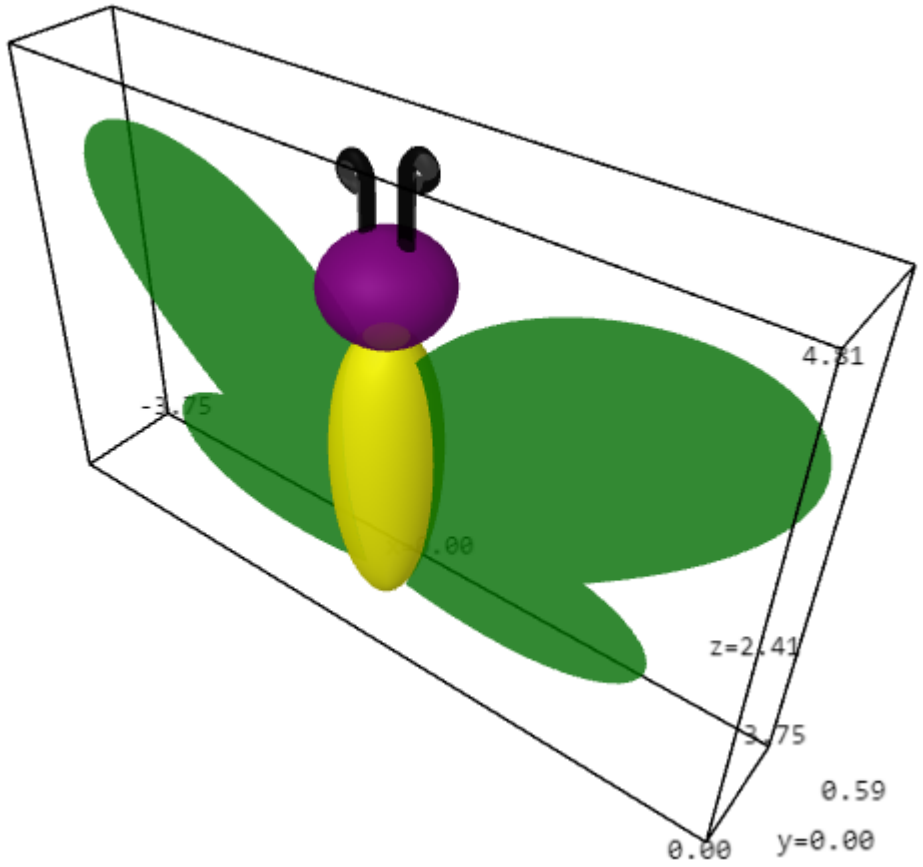
In [3]:
```
dibujar_superficie_parametrica(S1,0,1,0,2*pi,"yellow")+dibujar_superficie_parametrica(S2,0,1,0,2*pi,"purple")+dibujar_superficie
```

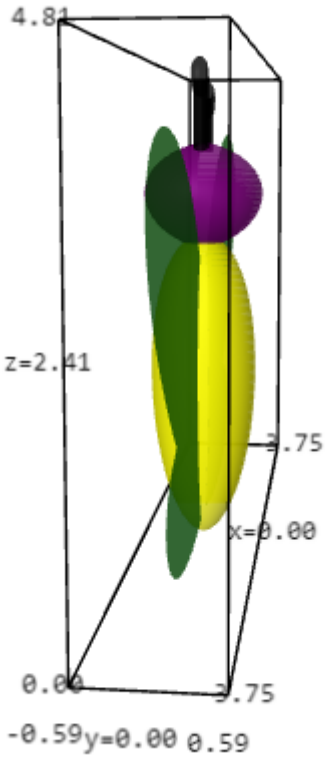
Out[3]:

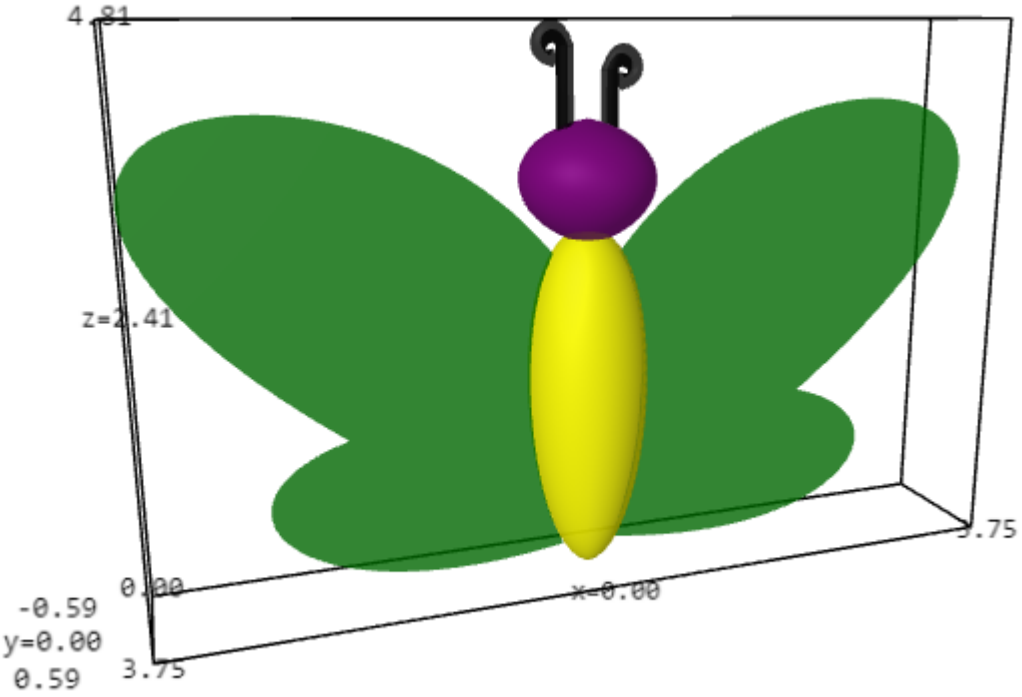



4.81
z=2.41
0.00
x=0.00
3.75
-0.59
y=0.00
0.59


4.81
z=2.41
0.00
-3.75


4.81
-3.75
.00
z=2.41
3.75
0.59
0.00
y=0.00


4.81
z=2.41
3.75
x=0.00
0.00
3.75
-0.59
y=0.00
0.59


4.81
z=2.41
0.00
-0.59
y=0.00
0.59
3.75
x=0.00
3.75