Construcción modelo 3D + Proyecciones

Auxiliar N°7

CC3501 – Modelación y Computación Gráfica para Ingenieros

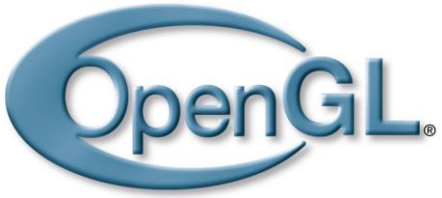
Contenidos de hoy

- Repaso estructura de un programa con OpenGL
- Modelar en OpenGL
- Proyecciones

Estructura con OpenGL

- Recordar que un programa básico consta de las siguientes partes:
  1) Importar librerías
  2) Definir constantes
  3) Iniciar la ventana
  4) Iniciar/construir los modelos
  5) Crear una cámara
  6) Iniciar el bucle de la aplicación
     1) Tickear el reloj
     2) Eliminar el búffer
     3) Ubicar la cámara
     4) Ver eventos
     5) Dibujar modelos (NO NECESARIAMENTE ESE ORDEN)

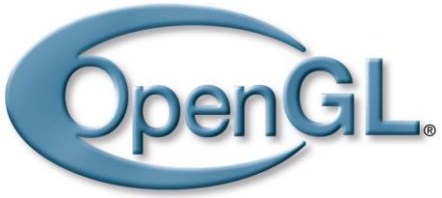
OpenGL – Gráficos en 3D



- Los vértices son la unidad básica de trabajo en OpenGL. Cada figura debe ser dibujada especificando previamente la posición de sus vértices.
- Funciones:

```
glVertex3f(x,y,z) # Recibe x,y,z float
glVertex3fv(v)    # Recibe un vector
glVertex3i(x,y,z) # Recibe x,y,z int
```

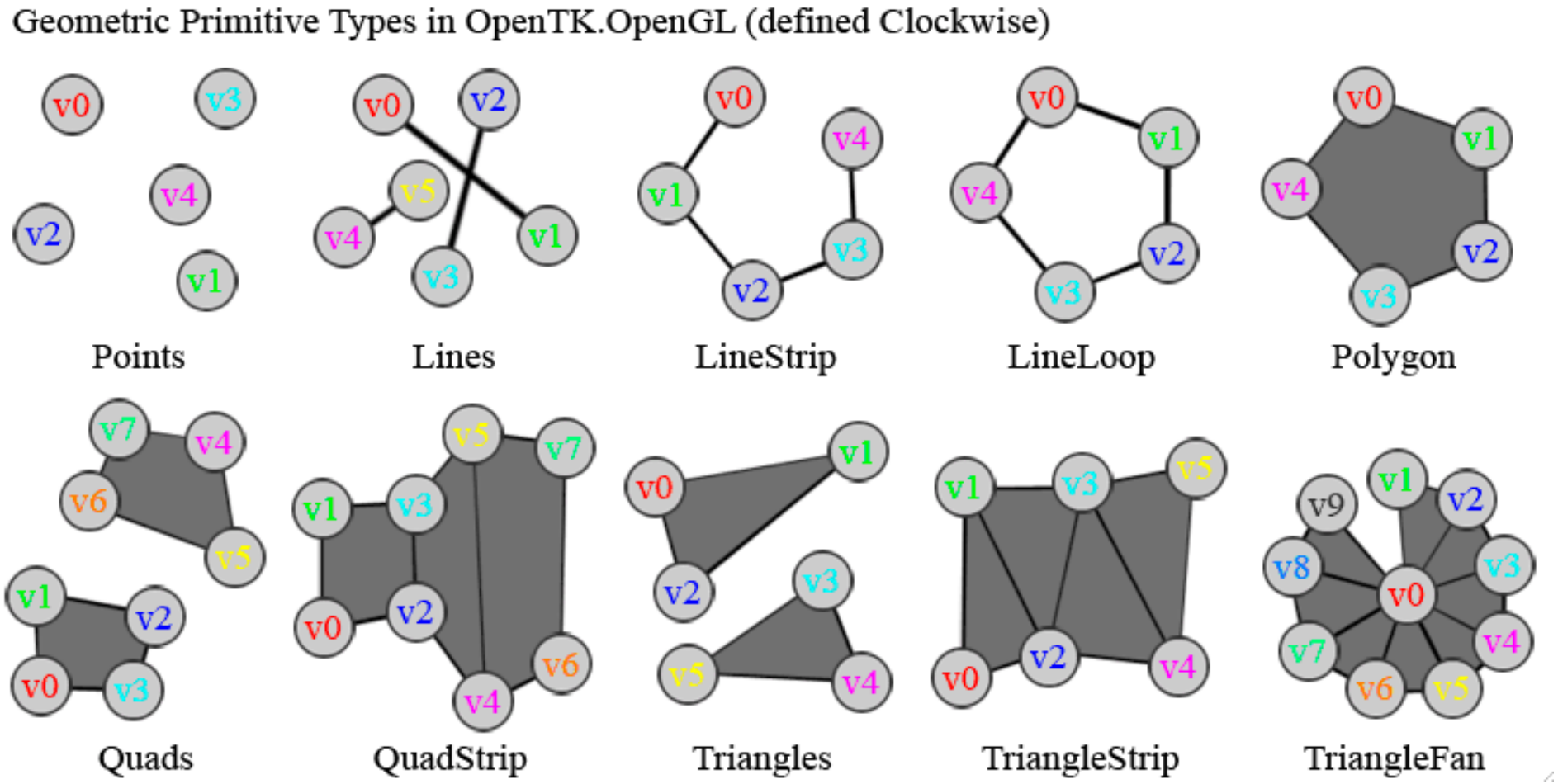
OpenGL – Primitivas



- Los vértices permiten dar una "guía" para posteriormente dibujar las primitivas.

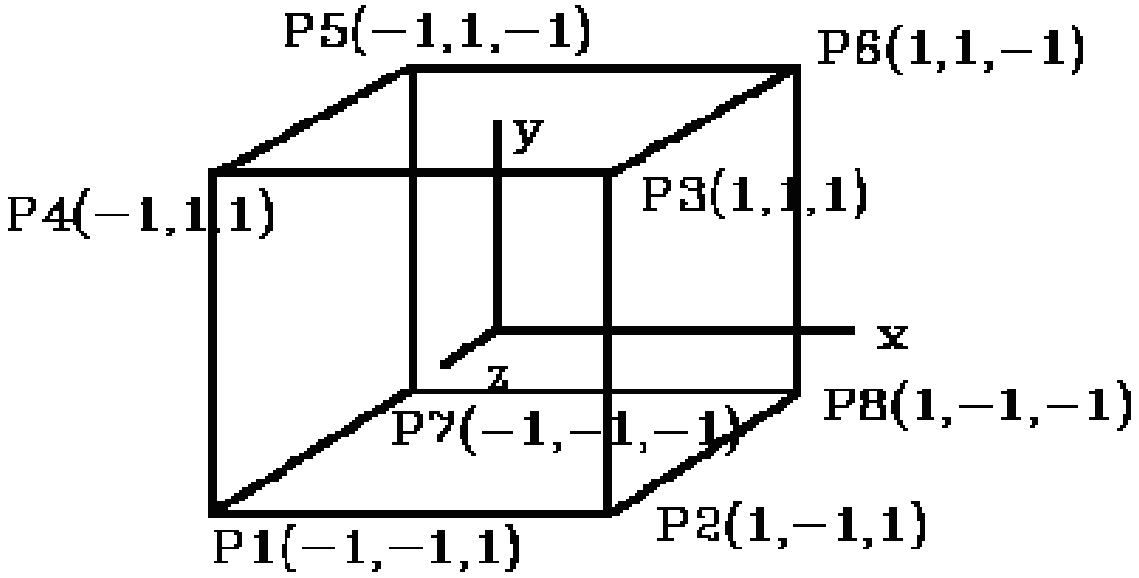
```
glBegin(PRIMITIVA) # Comienza a dibujar
glEnd() # Termina de dibujar
```

OpenGL – Primitivas



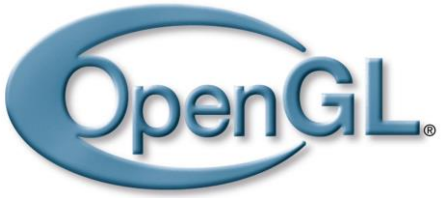
OpenGL – Ejemplo cubo 3D primitiva

```
p7 = Point3(-1.0, -1.0, -1.0)
p8 = Point3(1.0, -1.0, -1.0)
p2 = Point3(1.0, -1.0, 1.0)
p1 = Point3(-1.0, -1.0, 1.0)
p5 = Point3(-1.0, 1.0, -1.0)
p6 = Point3(1.0, 1.0, -1.0)
p3 = Point3(1.0, 1.0, 1.0)
p4 = Point3(-1.0, 1.0, 1.0)

obj = _gl.glGenLists(1)
_gl.glNewList(obj, _gl.GL_COMPILE)
_gl.glPushMatrix()
_gl.glBegin(_gl.GL_QUADS)
dibujar_vertices([p7, p8, p2, p1])
dibujar_vertices([p8, p6, p3, p2])
…
_gl.glEnd()
_gl.glPopMatrix()
_gl.glEndList()
```



dibujar_vertices([a,b,c,d]) dibuja en orden los vértices con glVertex3f(x,y,z)

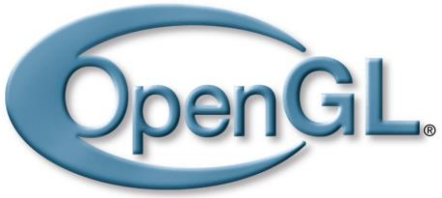
OpenGL – Transformaciones



- También es posible aplicar transformaciones geométricas a las figuras (listas de vértices).
  - Rotación
  - Traslación
  - Escalamiento

- OpenGL funciona con un sistema de matrices para todas las operaciones.

OpenGL – Transformaciones



- Para aplicar dichas transformaciones sólo basta usar las funciones de la api de OpenGL:
- Traslación:

```
glTranslatef(x,y,z)
```

- Escalamiento:

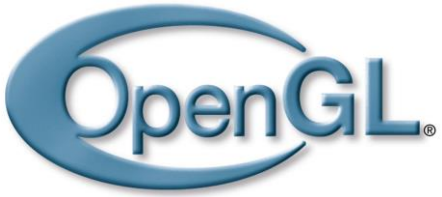
```
glScalef(x,y,z)
```

  - Aplica una transformación de escala con los factores x,y,z en cada eje. Valores negativos producen reflexiones en torno al eje.
- Rotación:

```
glRotate(angle,x,y,z)
```

  - Aplica una transformación de rotación de "angle" grados en torno al eje (x,y,z) Rotación según la regla de la mano derecha, ángulo en contra las manecillas del reloj. Se rota siempre respecto al origen.

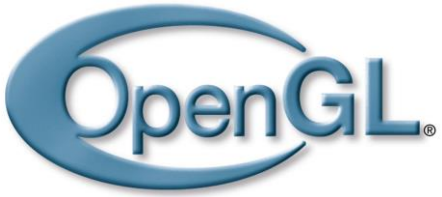
OpenGL – Matriz de transformación



- Cada transformación se aplica sobre la actual matriz de transformación del modelo, que afecta a toda la escena. (Multiplicación de matrices).
- Para aplicar transformaciones locales (no a toda la escena) se debe guardar la matriz actual, aplicar transformaciones y volver al estado anterior.
- Uso de stack de matrices. Funciones glPushMatrix() (guardar la matriz) y glPopMatrix() (restaurar la matriz).

- NOTA: Aplicar transformaciones en orden inverso.

OpenGL – Crear un modelo



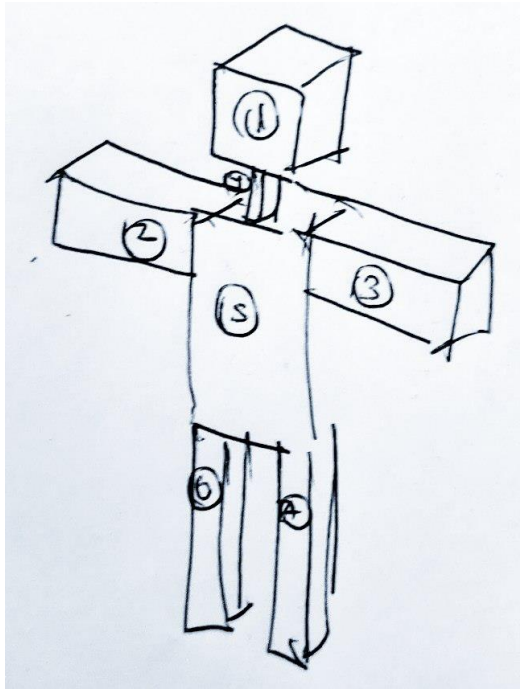
- Es recomendable crear listas, la cual contendrá un paquete de funciones OpenGL la cual puede ser posteriormente llamada en tiempo de ejecución.
- Dentro de esta lista se llama a primitivas, aplica colores, etc. Con tal de poder definir el modelo.
- Una manera fácil de crear un modelo es definirla a partir de una combinación de primitivas.

OpenGL – Crear un modelo

- Meta: Dibujar un T-POSE

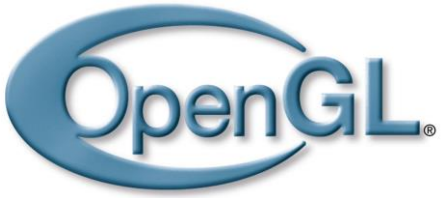
OpenGL – Crear un modelo

- Meta: ¿Cómo lo modelamos? Una combinación de primitivas, cubos en 3d. El objetivo es trasladarlos y escalarlos con tal de generar el modelo completo.



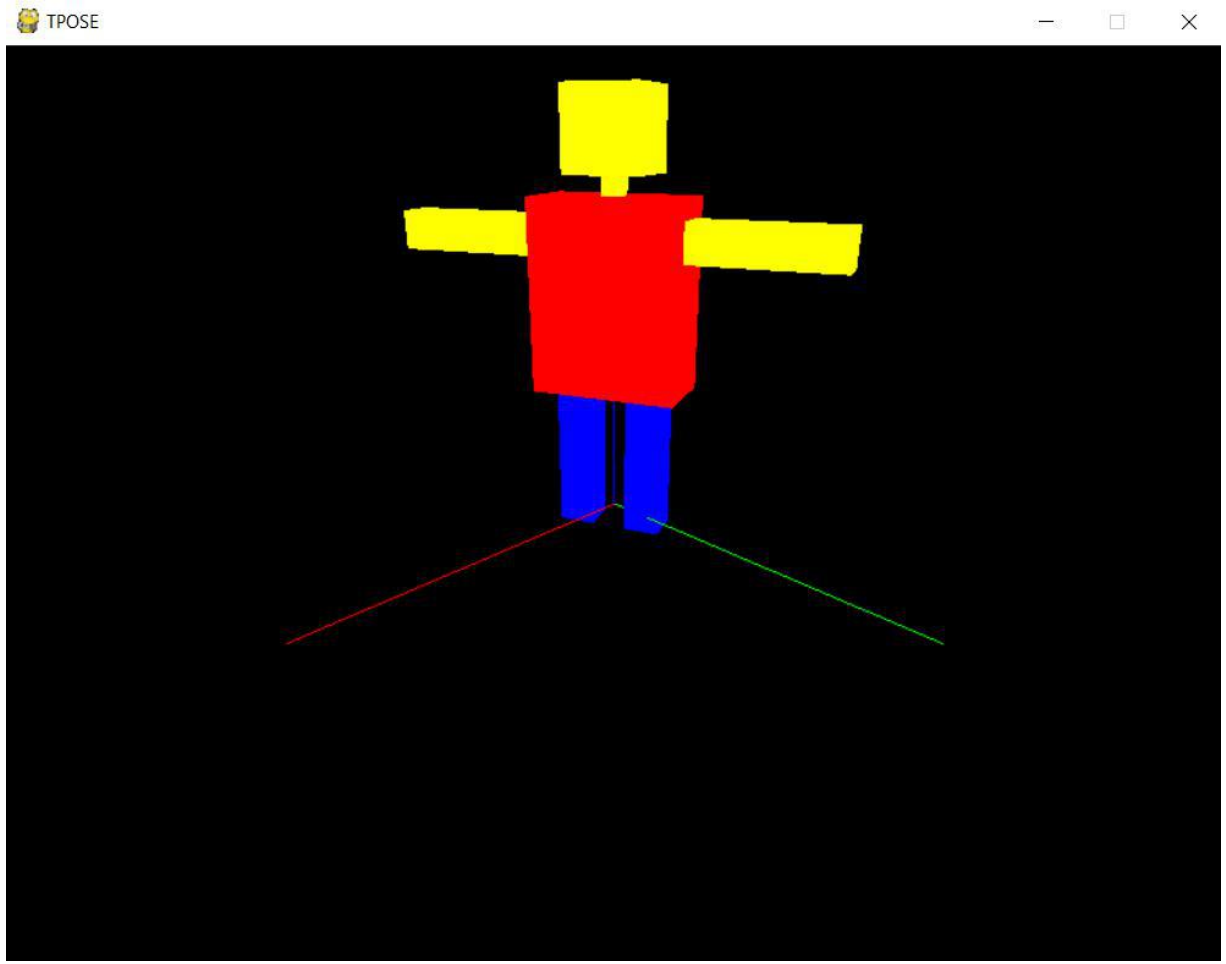
A TPOSE lo podemos representar con un total de 7 elementos, uno para la cabeza, otro para el cuello, dos para los brazos, uno para el torso y dos para las piernas

OpenGL – Crear un modelo

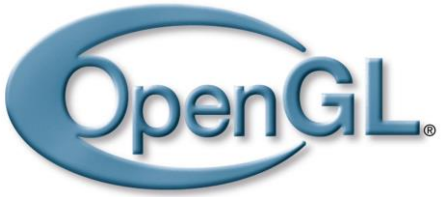



```
# Generamos un cubo (1x1x1)
cubo = create_cube()

# Creamos a la persona
tpose = glGenLists(1)  # Inicia una lista
glNewList(tpose, GL_COMPILE)

# La cabeza
glPushMatrix()
glTranslate(0, 0, 10)
glColor4fv([1, 1, 0, 1])
glCallList(cubo)
glPopMatrix()
…
glEndList()

glCallList(tpose) -> Se llaman a las funciones de OpenGL contenidas dentro de la lista
```

OpenGL – Crear un modelo

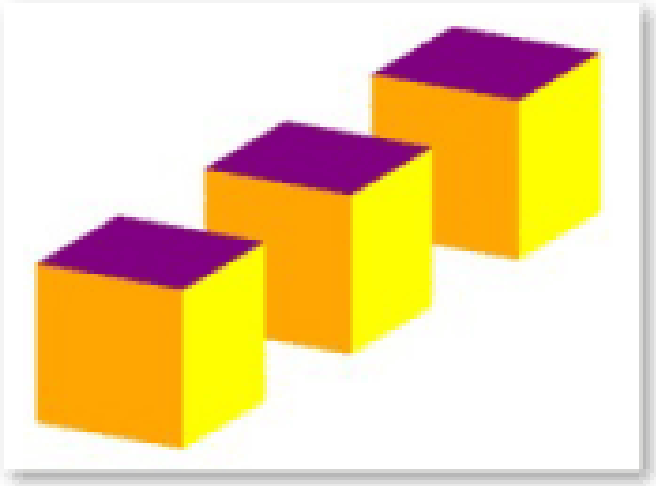


OpenGL – Proyecciones

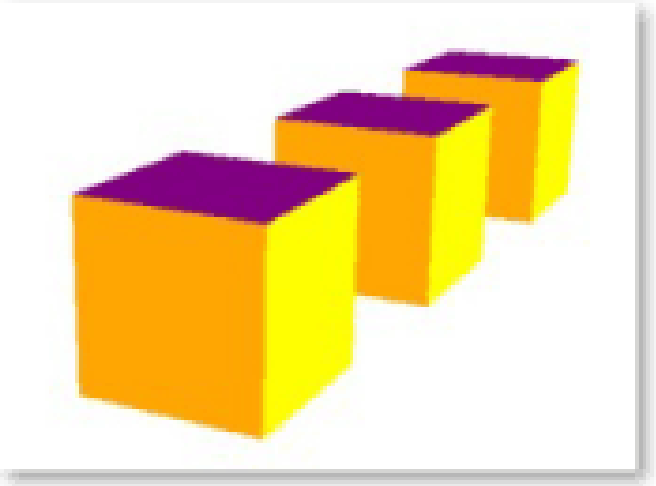


- Definen la manera en cómo se visualiza la escena.
- Dos alternativas:
  - En perspectiva
  - Ortográfica

Proyección ortográfica

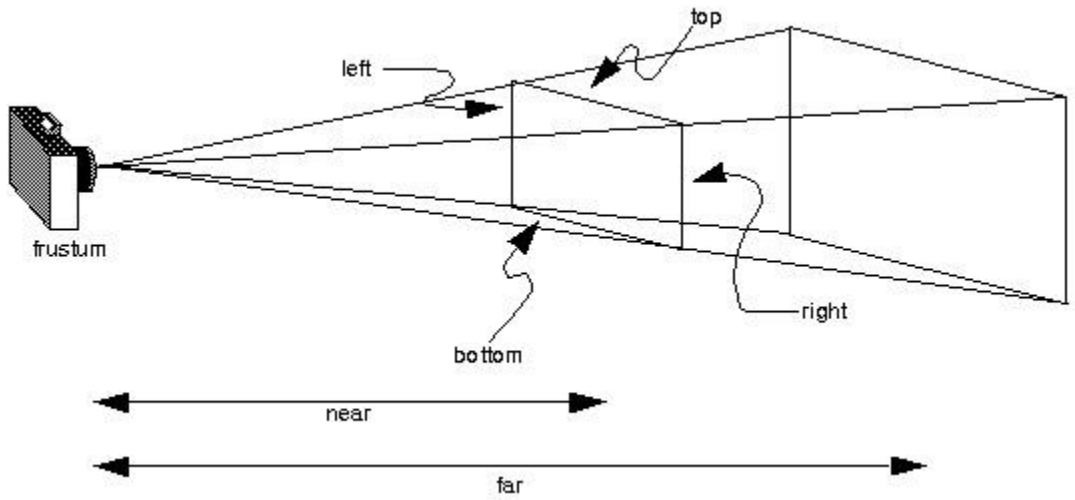


Proyección perspectiva

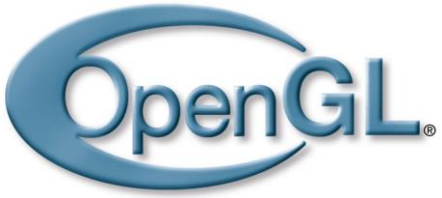


OpenGL – Proyecciones

- Perspectiva, limitada por una pirámide de visualización



OpenGL – Proyecciones

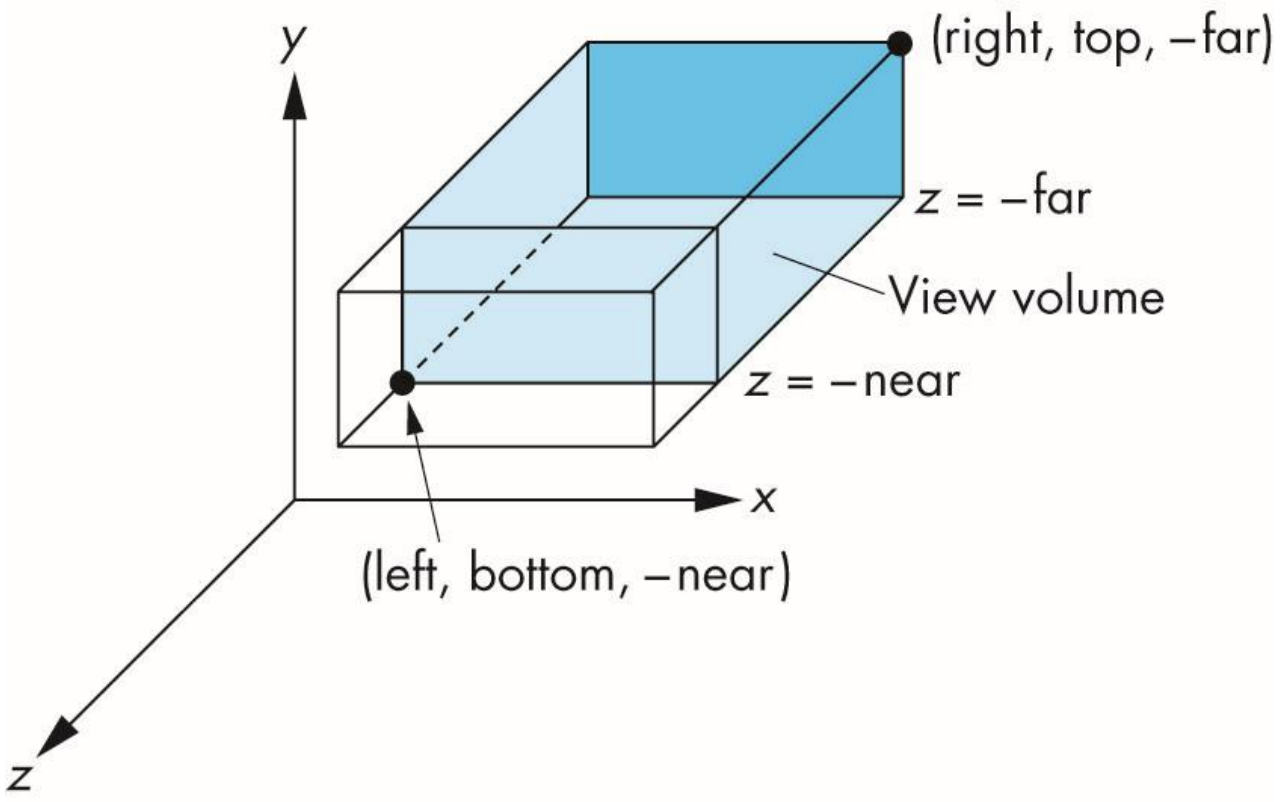


- Perspectiva, limitada por una pirámide de visualización

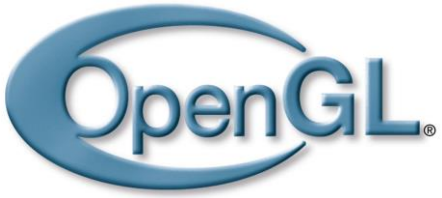
```
glLoadIdentity()

# Crea el viewport
glViewport(0, 0, int(w), int(h))
glMatrixMode(GL_PROJECTION)

# Proyección en perspectiva
gluPerspective(fov, float(w) / float(h), near, far)

# Setea el modo de la cámara
glMatrixMode(GL_MODELVIEW)
glLoadIdentity()
```

OpenGL – Proyecciones

- Ortográfica



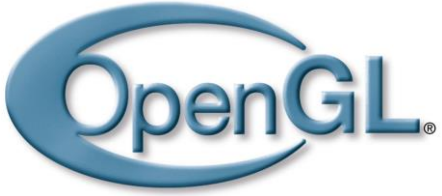
OpenGL – Proyecciones



- Ortográfica

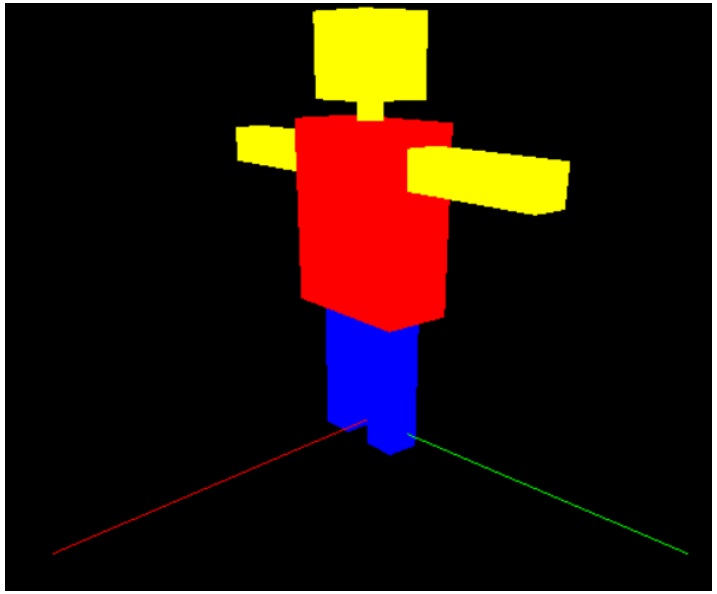
```
glLoadIdentity()

# Crea el viewport
glViewport(0, 0, int(w), int(h))
glMatrixMode(GL_PROJECTION)

# Proyección ortográfica
glOrtho(left, right, bottom, top, near, far)

# Setea el modo de la cámara
glMatrixMode(GL_MODELVIEW)
glLoadIdentity()
```

OpenGL – Proyecciones

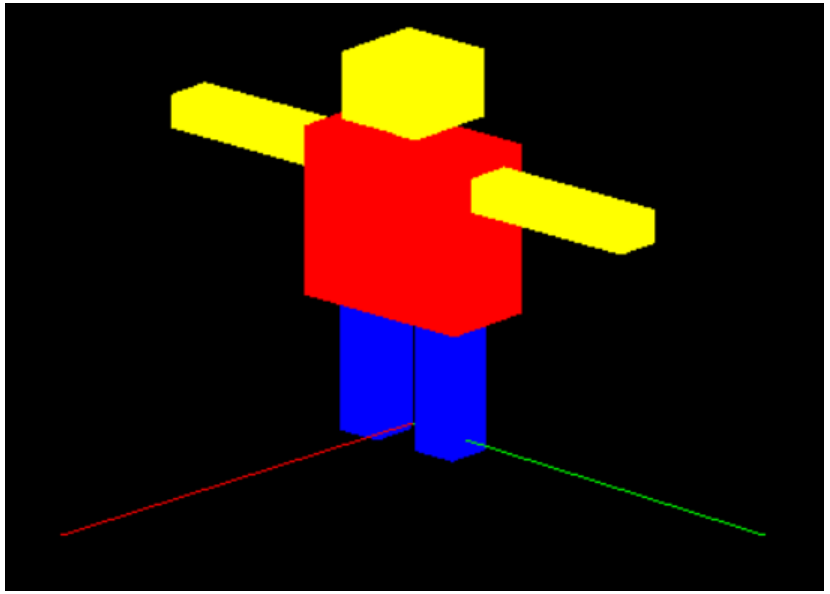


- Comparación

Perspectiva



Ortográfica



¿Consultas?