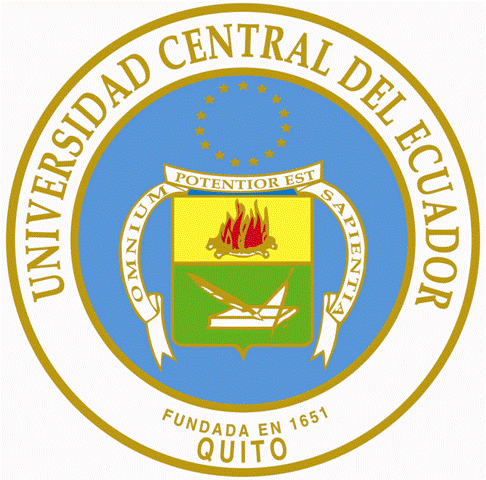
**UNIVERSIDAD CENTRAL DEL ECUADOR**

**FACULTAD DE INGENIERÍA CIENCIAS FÍSICAS Y MATEMÁTICAS**

**INGENIERÍA EN COMPUTACIÓN GRÁFICA**

**PROGRAMACIÓN GRÁFICA I**



**CONSULTA N#4**

* **MANEJO DE PILAS EN LA MATRIZ MODEL VIEW: PUSH Y POP**
* **CARGAR MATRIZ IDENTIDAD Y EL POR QUÉ UTILIZAR**
* **QUE VA PRIMERO: GL\_PROJECTION O GL\_MODELVIEW**

**ESTUDIANTE:**

**NICOLALDE PERUGACHI JOANA ESTEFANIA**

**DOCENTE:**

**JUAN PABLO GUEVARA GORDILLO**

**FECHA:**

**05/11/2019**

**Contenido**

[Manejo de Pilas en la matriz Model-View 3](#_Toc23926041)

[Matriz Identidad y su importancia 4](#_Toc23926042)

[¿Qué va primero GL\_PROJECTION O GL\_MODELVIEW? 5](#_Toc23926043)

[CONCLUSIONES 6](#_Toc23926044)

[RECOMENDACIONES 6](#_Toc23926045)

[BIBLIOGRAFÍA 6](#_Toc23926046)

**Tabla de Imágenes**

[**Imagen 1.** Ejemplo Push-Pop en Traslación 4](#_Toc23969281)

[**Imagen 2.** Pipeline Gráfico 6](#_Toc23969282)

# Manejo de Pilas en la matriz Model-View

Ya se conoce la utilización de primitivas en OpenGL, ahora nuestro estudio se basa en aprender acerca de transformaciones geométricas, las cuales nos permiten rotar un objeto, trasladarlo a otra posición e incluso cambiarlo de tamaño, a estas las conocemos como ROTACIÓN, TRASLACIÓN Y ESCALAMIENTO respectivamente. Debemos saber que este tipo de transformaciones se aplican antes de que un objeto sea visualizado (García E. , s.f.).

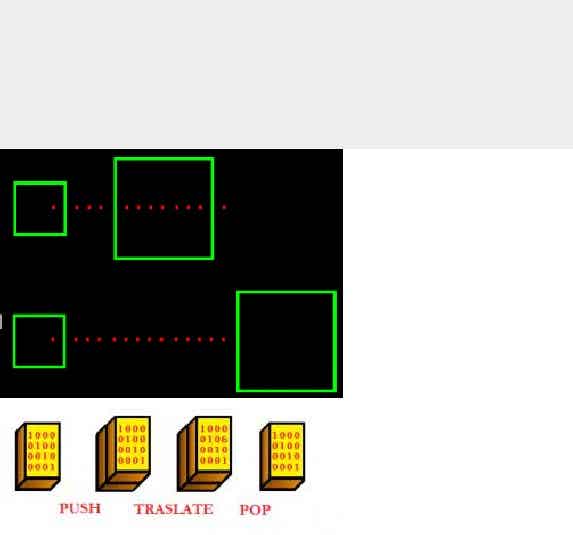
En el caso de la Traslación y el Escalado, aunque se escriba en el código de manera arbitraria, el comportamiento de la pila exige que la primera transformación será el Escalado, pues este será el primer procedimiento en entrar en la pila, posteriormente continuará la rotación y finalmente la traslación (García E. , s.f.).

García refiere que OpenGL cuenta con una pila destinada para las transformaciones geométricas y de la cámara (GL\_MODELVIEW) y otra para proyecciones (GL\_PROJECTION). Es necesario indicar en que pila estamos trabajando y para ello se usa la función que activa la matriz de transformación con el nombre de la pila glMatrixMode(Nombre\_Pila) (García, s.f.).

La matriz de transformación Model-View se entiende como una pila, en la que en cada transformación que se agrega entra a la pila como la última y al salir será la primera a esto conocemos como LIFO, de esta manera podemos salvar el estado de la pila y recuperarlo en cualquier momento (García & Guevara, 2004).

Cuando se trata de sólo realizar una transformación, no hay tantos problemas como cuando se desea aplicar varias transformaciones a diversos objetos, pues todos estos procesos se irán almacenando en la pila, y para poder efectuar esto, se debe mantener un control de modificación de contenido de pila, para ellos contamos con tres funciones importantes: glLoadIdenty(), glPushMatrix() y glPopMatrix() (García E. , s.f.).

A continuación, la imagen 1 que muestra gráficamente un proceso de traslación con la pila.



**Imagen 1.** Ejemplo Push-Pop en Traslación

Fuente: (García, s.f.)

* **glLoadIdenty()**

Esta función sustituye el contenido de la pila por la matriz de identidad (García E. , s.f.).

* **glPushMatrix()**

Esta función en cambio, realiza una copia de la matriz superior y la coloca encima de la pila, de manera que ambas matrices superiores sean iguales, es decir duplica la matriz y así cualquier transformación que se aplique sobre un objeto se realizará sólo a la matriz superior de la pila, manteniendo los valores intactos de la anterior, como los tenía antes de llamarla (García E. , s.f.).

* **glPopMatrix()**

Esta función por otro lado elimina la matriz superior creada anteriormente, de forma que en la parte superior de la pila únicamente quede la matriz previa a la llamada de la función glPushMatrix() (García E. , s.f.).

# Matriz Identidad y su importancia

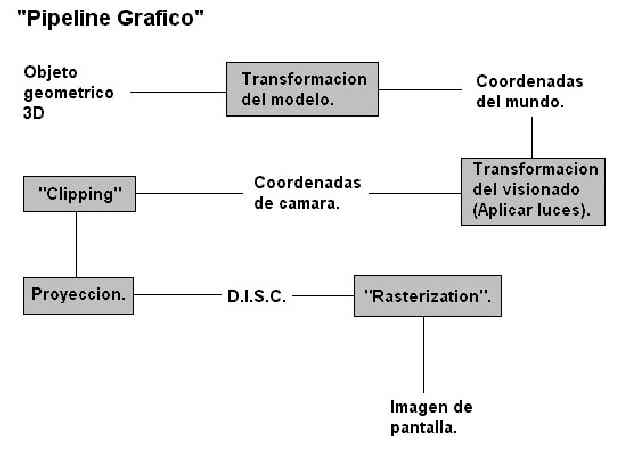
Lo primero que se debe hacer es inicializar la matriz, esto lo hacemos con la matriz de identidad, la cual es el elemento neutro de la multiplicación de matrices, de esta manera se puede asegurar que se limpia por completo antes de aplicar cualquier transformación; en el caso que no se hiciera esto, se puede correr el riesgo de añadir transformaciones a otras ya existentes y así, los resultados obtenidos no serán los esperados (García & Guevara, 2004).

Como ya se mencionó anteriormente la función **glLoadIdenty()** es la que permite reemplazar el contenido de la pila por la matriz de identidad.

# ¿Qué va primero GL\_PROJECTION O GL\_MODELVIEW?

Las transformaciones de Model View se ejecutan antes que las transformaciones de proyección, pues la transformación de visualización es la encargada de trasladar, rotar, escalar e incluso torcer cualquier objeto para que sea dibujado en pantalla tal y como debe estar dispuesto en el mundo, mientras que la de proyección permite pasar de coordenadas 3D del mundo a coordenadas 2D de nuestro plano de proyección, una vez ya transformado y visualizado el modelo (García & Guevara, 2004).

Una de las pautas en las que se puede mostrar el funcionamiento de OpenGL, inclusive un proceso para programar, es el pipeline gráfico, como se muestra a continuación en la imagen 2.



**Imagen 2.** Pipeline Gráfico

Fuente: (García & Guevara, 2004)

Ambas se concatenan y de cualquier manera OpenGL funciona independientemente del orden del pipeline gráfico, no es necesario cambiarlos si se cambia de plataforma, pues siempre funcionará de la misma manera, ya que la librería de por sí lo aplicará de manera apropiada. Lo único es que de acuerdo al sistema se conseguirá mayor o menor velocidad (García & Guevara, 2004).

# CONCLUSIONES

* En conclusión, OpenGL tiene un orden en las transformaciones, primero el escalamiento, luego rotación y finalmente traslación.
* Las pilas utilizadas son de tipo LIFO, es decir ultimas en entrar y primeras en salir, y estas nos sirven para mantener un control en su contenido y así mantener a salvo el procedimiento original.
* La función glPushMatrix(), duplica la matriz original y la coloca en la parte superior de la pila, mientras que glPopMatrix () elimina esta nueva matriz, dejando los valores intactos como antes de modificarla.
* La función glLoadIdenty() es el elemento neutro de la multiplicación y es la que permite resetear antes de empezar las operaciones, pues si no hacemos esto pueden tomarse valores de otras transformaciones, y no lograr resultados adecuados.
* Es necesario primero encargarnos de la visualización de un modelo y sus operaciones, luego proyectarlo pasando de unas coordenadas a otras, es decir primero se ejecuta GL\_MODELVIEW y luego GL\_PROJECTION.

# RECOMENDACIONES

* Se recomienda aplicar las diferentes transformaciones ya utilizando las funciones glPushMatrix() y glPopMatrix().
* También se recomienda siempre cargar la matriz identidad antes de realizar cualquier transformación, con glLoadIdenty().
* Tratar de respetar el orden en las transformaciones, escalamiento, rotación y finalmente traslación, para mejor optimización de la velocidad del sistema, aunque de por si OpenGL, lo hace automáticamente independientemente del orden en que las coloquemos.

# BIBLIOGRAFÍA

García, E. (s.f.). *Transformaciones Geométricas en OpenGL.* PAPER. Recuperado el 05 de Novimebre de 2019, de https://www.academia.edu/29096393/Transformaciones\_Geom%C3%A9tricas\_en\_OpenGL

García, O., & Guevara, A. (2004). *Introducción a la Programación Gráfica con OpenGL.* Escuela Técnica Superior de Ingeniería Electrónica eInformática La Salle.