

Hola mundo en OpenGL + GIT

Auxiliar N°4

CC3501 – Modelación y Computación Gráfica para Ingenieros

Contenidos de hoy

- GIT: Breve tutorial
- OpenGL
- Pygame
- Ejemplo pequeño: dibujar un cuadrado con colores

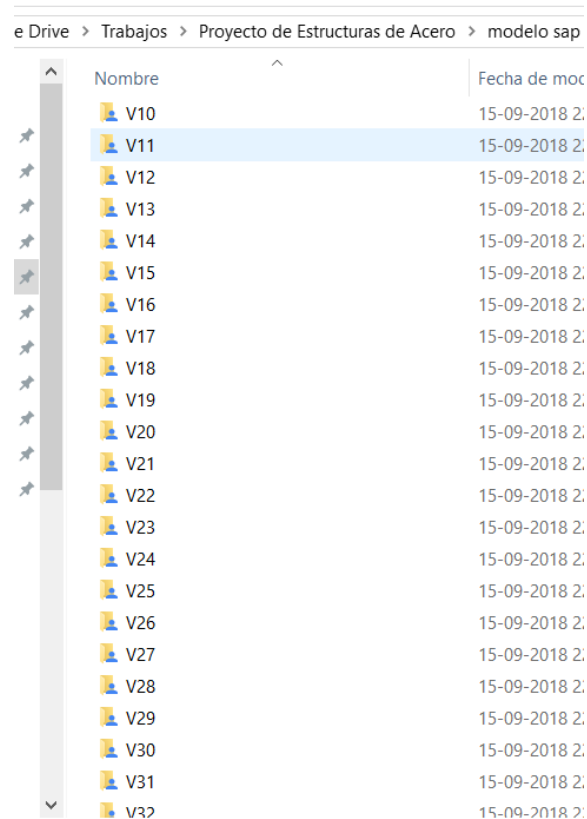
Git – Mini tutorial

- ¿Qué es el **control de versiones**, y por qué debería importarte? El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.
- Un método de control de versiones usado por mucha gente es copiar los archivos a otro directorio.

Fuente: <https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>

Git - Introducción

- Problemas de copiar archivos a un directorio distinto:



e Drive > Trabajos > Proyecto de Estructuras de Acero > modelo sap		
Nombre	Fecha de mod	
V10	15-09-2018 2:	
V11	15-09-2018 2:	
V12	15-09-2018 2:	
V13	15-09-2018 2:	
V14	15-09-2018 2:	
V15	15-09-2018 2:	
V16	15-09-2018 2:	
V17	15-09-2018 2:	
V18	15-09-2018 2:	
V19	15-09-2018 2:	
V20	15-09-2018 2:	
V21	15-09-2018 2:	
V22	15-09-2018 2:	
V23	15-09-2018 2:	
V24	15-09-2018 2:	
V25	15-09-2018 2:	
V26	15-09-2018 2:	
V27	15-09-2018 2:	
V28	15-09-2018 2:	
V29	15-09-2018 2:	
V30	15-09-2018 2:	
V31	15-09-2018 2:	
V32	15-09-2018 2:	

- ¿Qué cambios hice en cada versión?
- ¿Qué archivos se cambiaron?
- (Pensando en un proyecto con varios colaboradores) ¿Quién hizo los cambios?
- ¿Cuándo se hicieron los cambios?
- ¿Qué pasa si el proyecto pesa mucho? ¿Conviene tener una carpeta completa por cada versión?

Git – La solución

- Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos.
- Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea.
- Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.

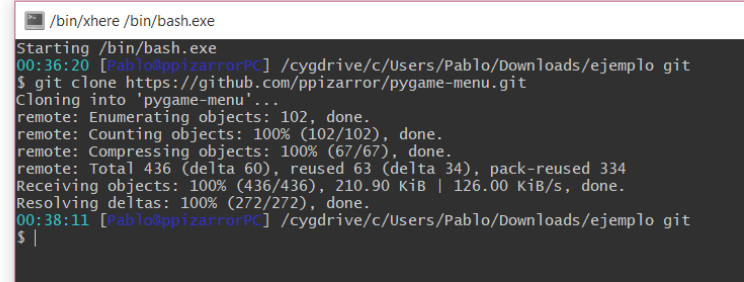
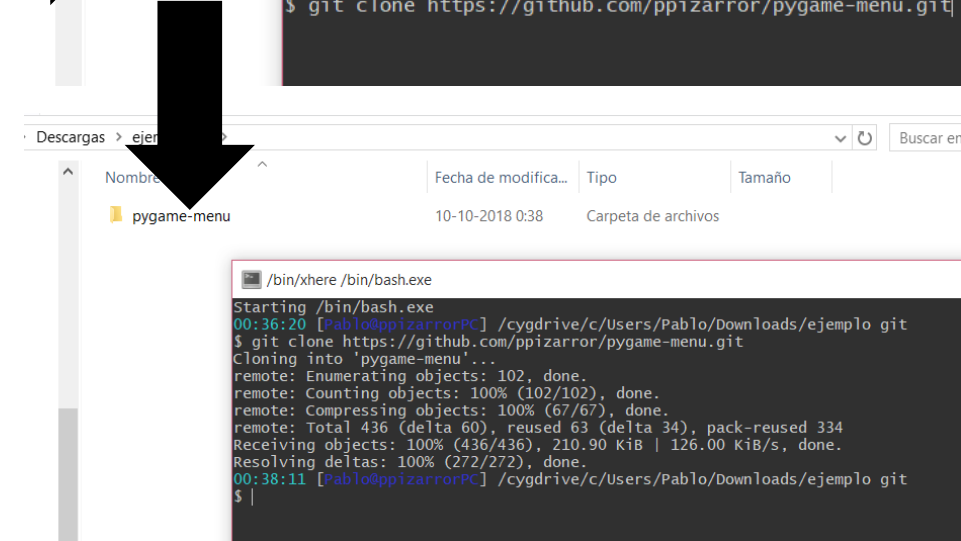
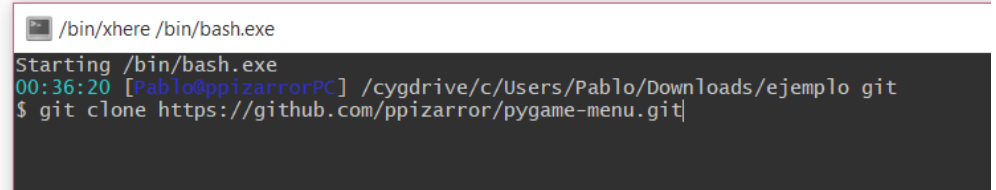
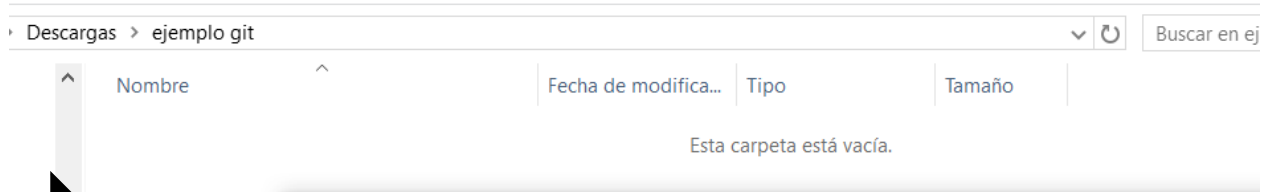
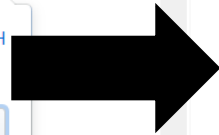
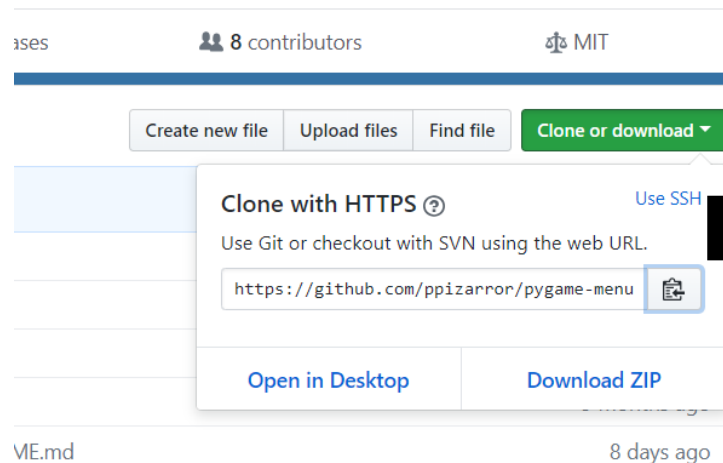
Fuente: <https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git>

Git – Creación de un repositorio

- Crea un directorio nuevo, ábrelo y ejecuta `git init` para crear un nuevo repositorio de git.
- También se puede crear un nuevo repositorio *remoto* en la nube, algunos servicios:
 - Github github.com/
 - Bitbucket <https://bitbucket.org>
 - Gitlab gitlab.com

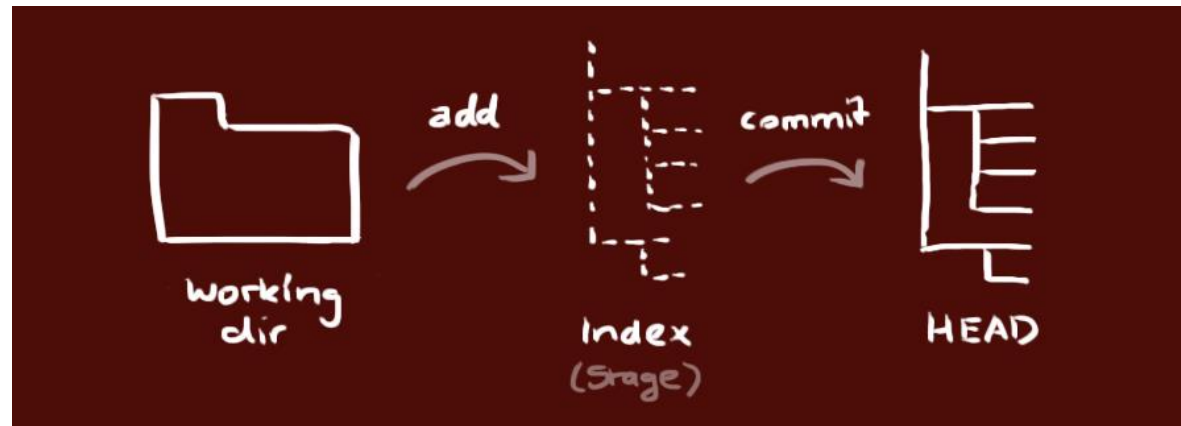
Git – Clonar un repositorio

- Crea una copia local del repositorio ejecutando `git clone /path/to/repository` desde una terminal.



Git – Flujo de trabajo

- Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu **Directorio de trabajo** que contiene los archivos, el segundo es el **Index** que actua como una zona intermedia, y el último es el **HEAD** que apunta al último commit realizado.

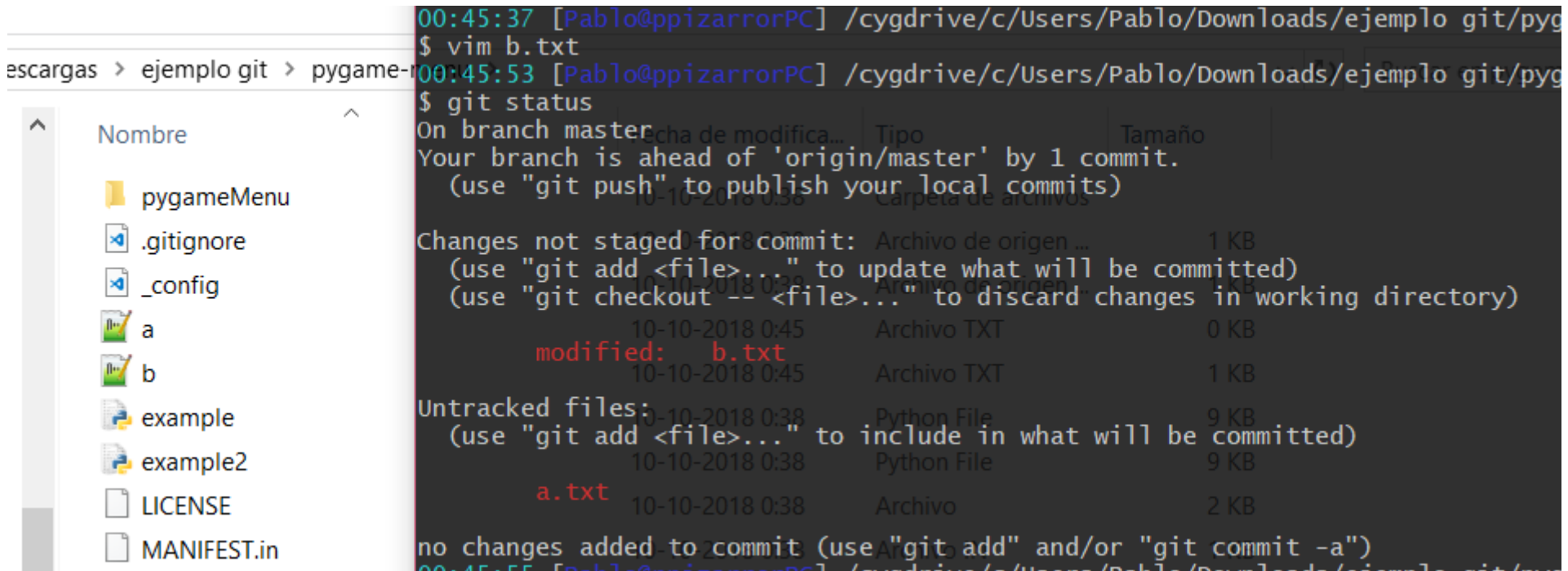


Git – Commits

- Un commit se puede entender como un paquete de cambios, en donde se pueden añadir archivos nuevos, agregar cambios en ciertos archivos o borrar archivos.
- Un commit tiene asociado un usuario, una fecha y un comentario.
- Para revisar qué cosas se han cambiado desde el último commit hasta la fecha se debe usar el comando `git status`

Git – Commits & status

- Ejemplo: Suponer que añadí un archivo `a.txt` y añadí tres líneas al archivo `b.txt`, `git status` dirá lo siguiente:



The screenshot shows a terminal window with the following commands and output:

```
00:45:37 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pyg
$ vim b.txt
00:45:53 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pyg
$ git status
```

The output of `git status` is as follows:

```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   b.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        a.txt

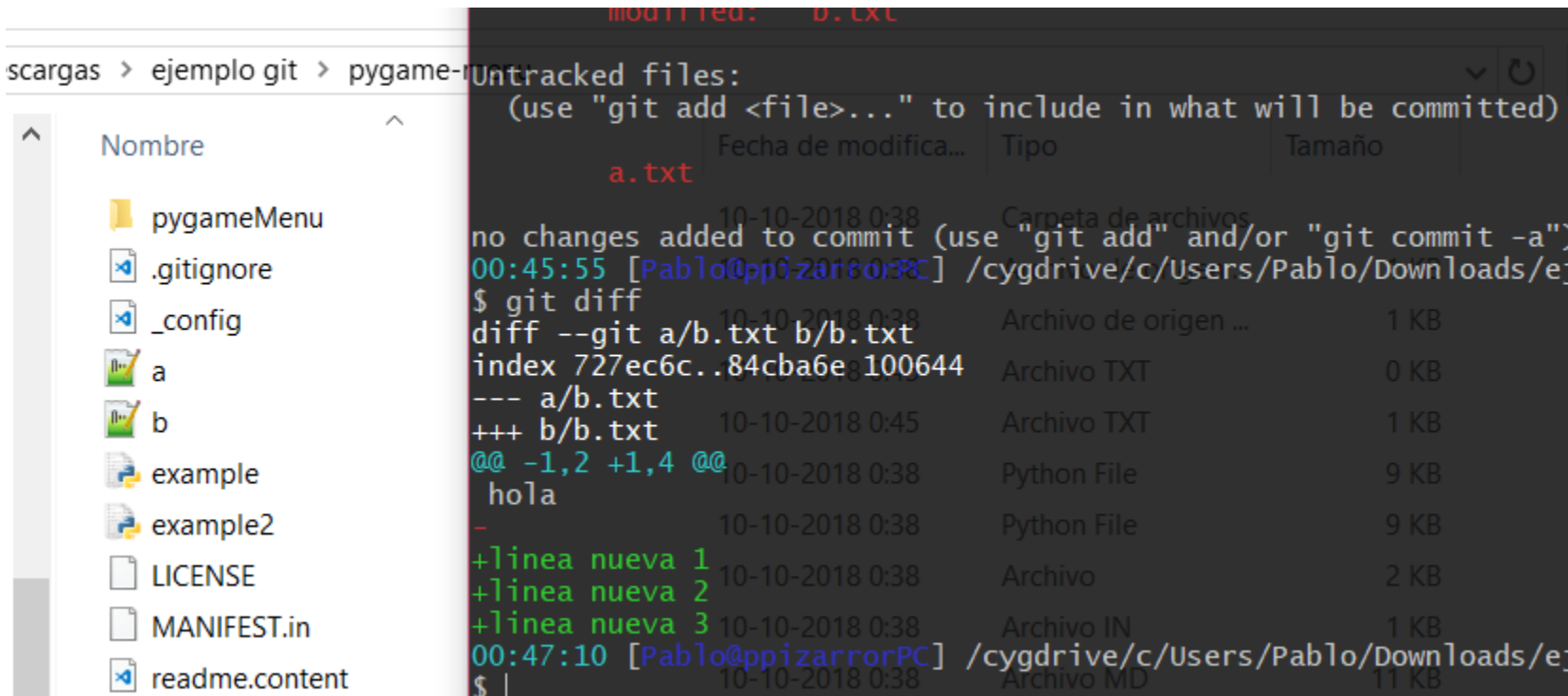
no changes added to commit (use "git add" and/or "git commit -a")
```

On the left side of the terminal, a file explorer sidebar is visible, showing the following files and folders:

- pygameMenu
- .gitignore
- _config
- a
- b
- example
- example2
- LICENSE
- MANIFEST.in

Git – Commits & status

- Ejemplo: Si quiero un detalle de los cambios nuevos se puede usar el comando `git diff`.



The image shows a file explorer window on the left and a terminal window on the right. The file explorer shows a directory named 'ejemplo git' containing a 'pygame-menu' folder and several files: '.gitignore', '_config', 'a', 'b', 'example', 'example2', 'LICENSE', 'MANIFEST.in', and 'readme.content'. The terminal window shows the output of the 'git diff' command, which displays the differences between the current state and the last commit. The output indicates that files 'a' and 'b' have been added, and file 'example' has been modified. The terminal also shows the command 'git diff' being executed and the resulting diff output.

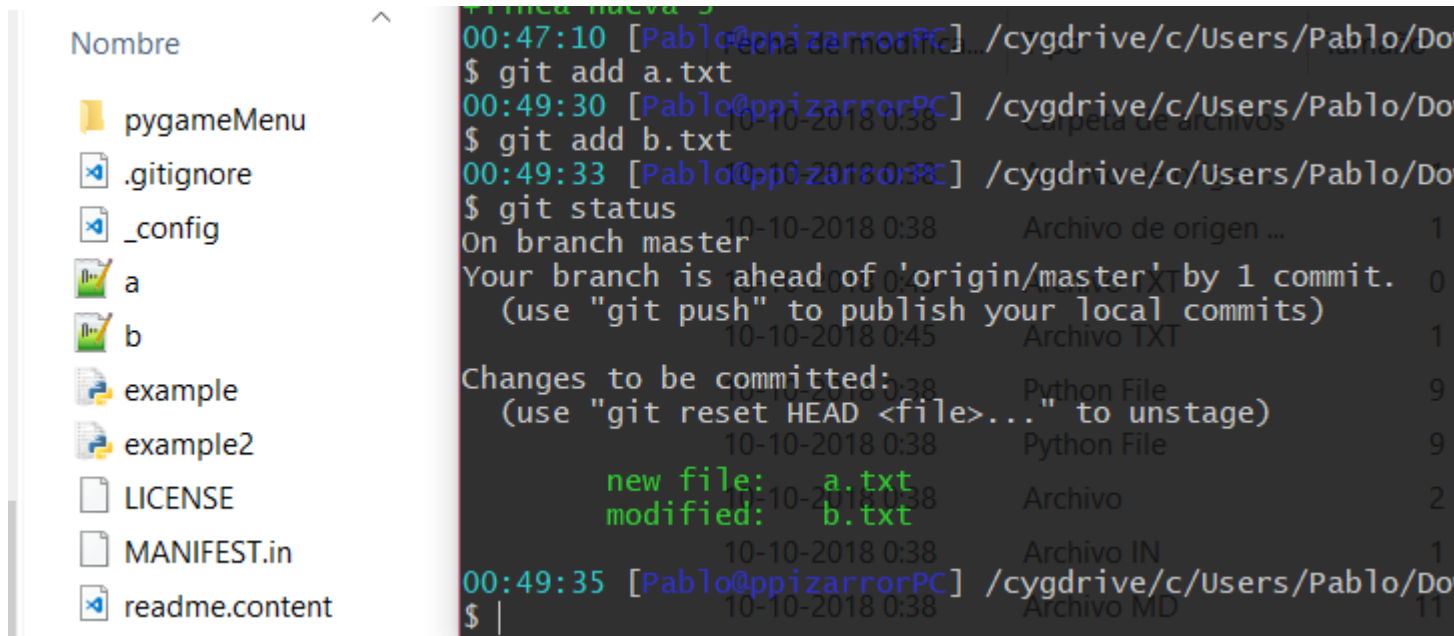
```
Untracked files:
(use "git add <file>..." to include in what will be committed)

a.txt
b.txt

no changes added to commit (use "git add" and/or "git commit -a")
00:45:55 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git
$ git diff
diff --git a/b.txt b/b.txt
index 727ec6c..84cba6e 100644
--- a/b.txt
+++ b/b.txt
@@ -1,2 +1,4 @@
 hola
-
10-10-2018 0:38 Python File 9 KB
+linea nueva 1
+linea nueva 2
+linea nueva 3
00:47:10 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git
$
```

Git – Commits & status

- Para añadir cambios al commit usar el comando `git add <filename>`, se pueden usar wildcards (*) o bien añadir todos los archivos nuevos/modificados con `git add --all`



The image shows a file explorer on the left and a terminal window on the right. The file explorer lists files: pygameMenu, .gitignore, _config, a, b, example, example2, LICENSE, MANIFEST.in, and readme.content. The terminal window shows the following commands and output:

```
00:47:10 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Do
$ git add a.txt
00:49:30 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Do
$ git add b.txt
00:49:33 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Do
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   a.txt
        modified:   b.txt
00:49:35 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Do
$ |
```

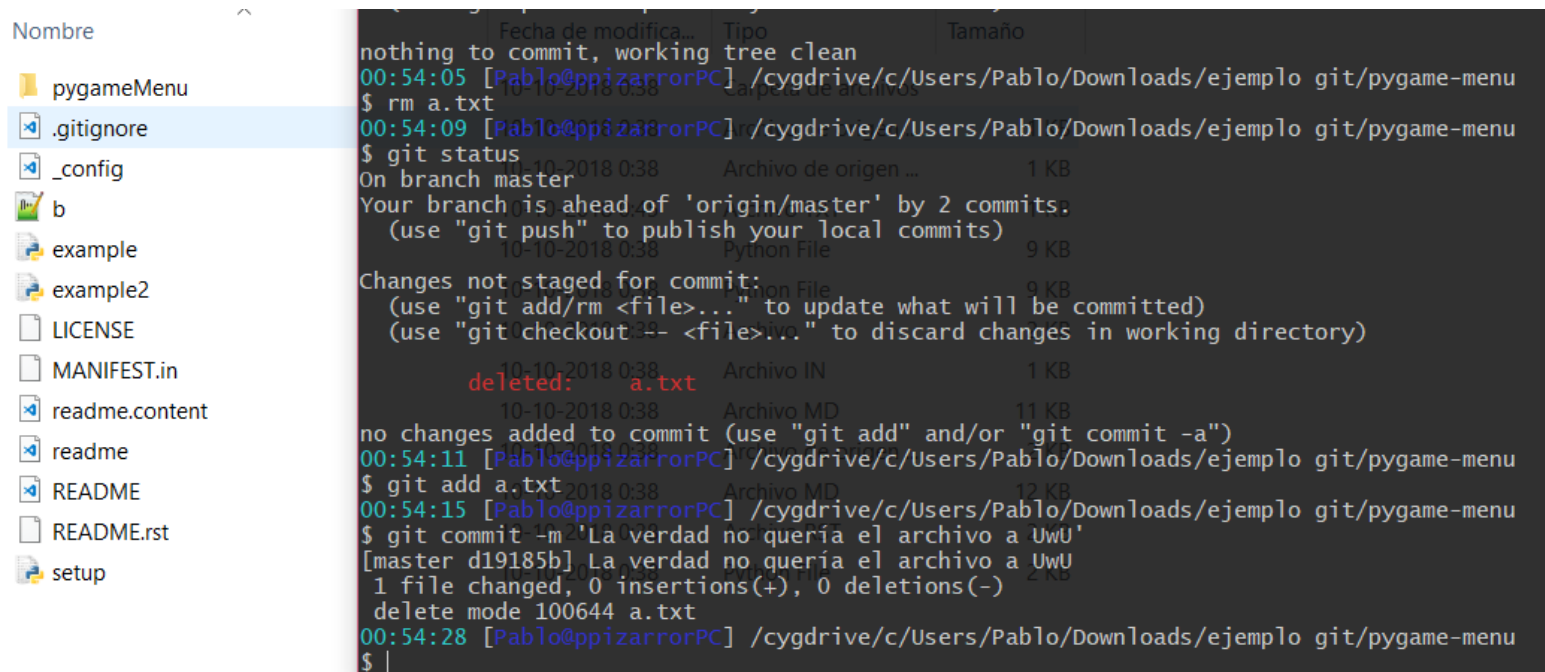
Lo que dice esta terminal es que el nuevo commit contiene un nuevo archivo a.txt y uno modificado b.txt

Git – Commits & status

- Para cerrar el commit se debe utilizar el comando `git commit -m "Commit message"` con esto se añade un comentario al paquete de cambios.

Git – Commits & status

- Se pueden hacer muchos commits seguidos, estos se mantendrán siempre de manera local (en tu computador) hasta que decidas subirlos al repositorio maestro (entiéndase por lo que está en github).



The image shows a file explorer on the left and a terminal window on the right. The file explorer displays a directory named 'pygameMenu' with files: '.gitignore', '_config', 'b', 'example', 'example2', 'LICENSE', 'MANIFEST.in', 'readme.content', 'readme', 'README', 'README.rst', and 'setup'. The terminal window shows the following commands and output:

```
nothing to commit, working tree clean
00:54:05 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pygame-menu
$ rm a.txt
00:54:09 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pygame-menu
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

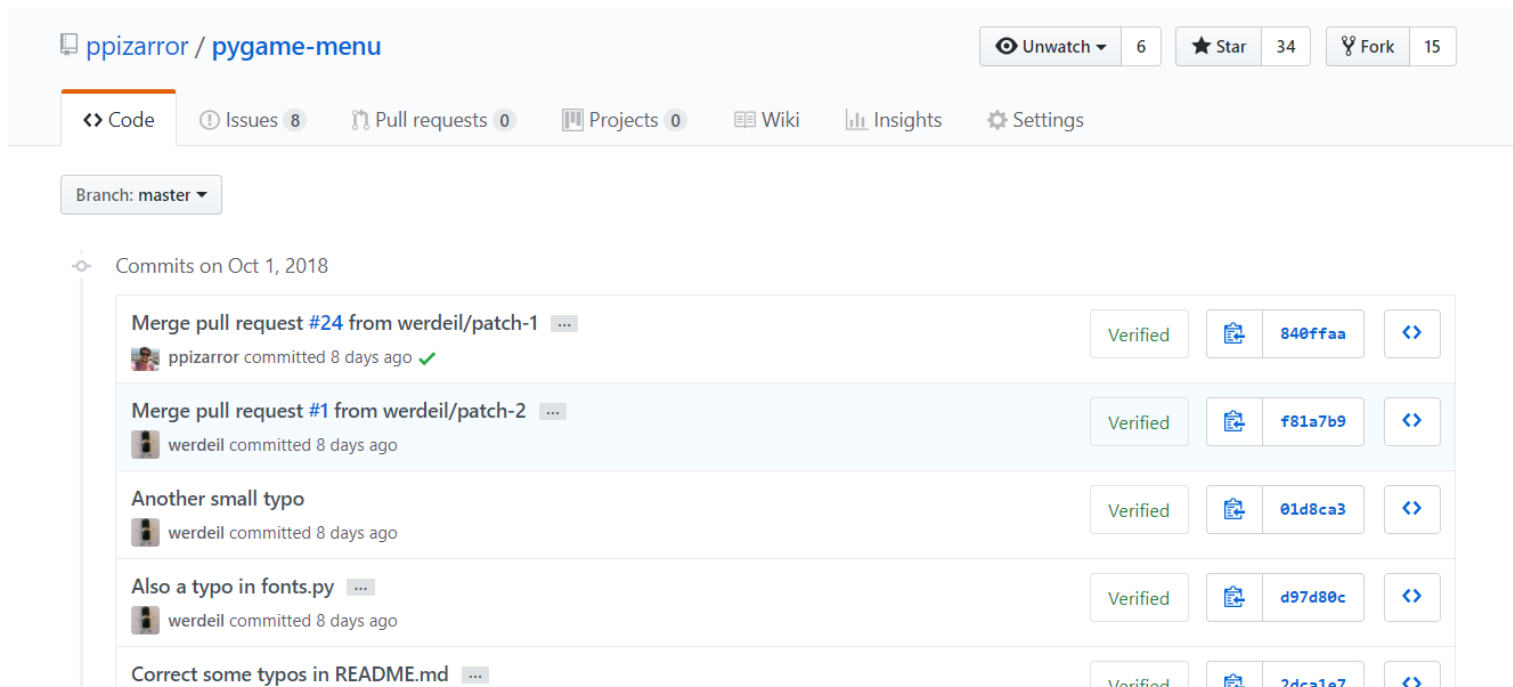
        deleted:    a.txt

no changes added to commit (use "git add" and/or "git commit -a")
00:54:11 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pygame-menu
$ git add a.txt
00:54:15 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pygame-menu
$ git commit -m "La verdad no queria el archivo a UwU"
[master d19185b] La verdad no queria el archivo a UwU
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 a.txt
00:54:28 [Pablo@ppizarrorPC] /cygdrive/c/Users/Pablo/Downloads/ejemplo git/pygame-menu
$
```

Se borró el archivo a.txt y se crea el commit.

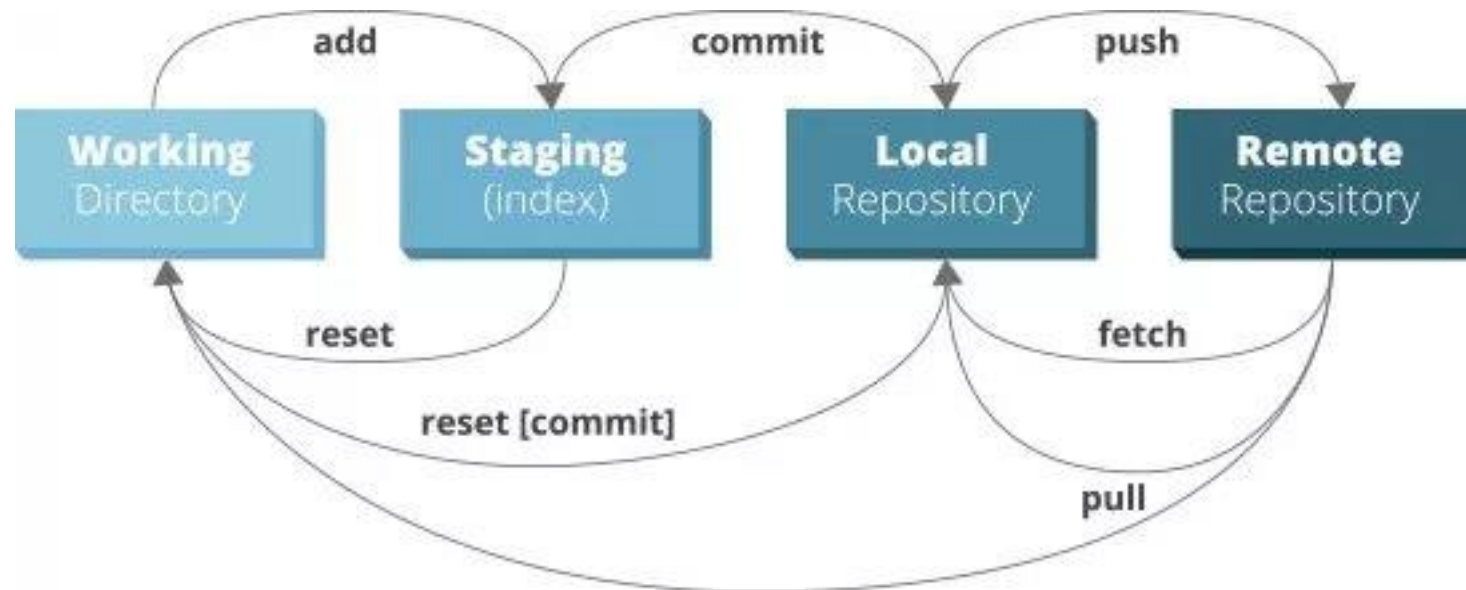
Git – Publicar los commits

- Para publicar los commits al repositorio maestro se debe utilizar **git push**, literalmente se empujan los nuevos cambios.
- En el repositorio se puede revisar todos los commits realizados.



Git – Revisar los commits

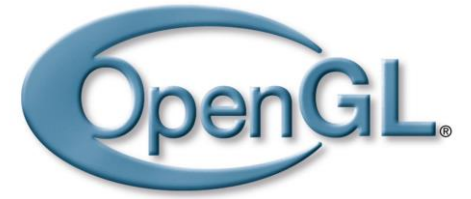
- Es posible descargar el proyecto en una fecha anterior a cada commit.
- Se puede observar cada cambios de los commits.
- Para descargar los commits a tu computador utilizar el comando `git pull`



Git – Descargar los commits

- Al realizar git pull es posible que se puedan generar conflictos (se modifica un archivo que ustedes también modificaron), para ello cada caso se debe revisar de manera manual.
- Cada archivo *conflictivo* se debe luego añadir a un nuevo commit con `git add <filename>`
- Eso sólo ocurre cuando se trabaja en un proyecto de varios colaboradores

OpenGL - Introducción



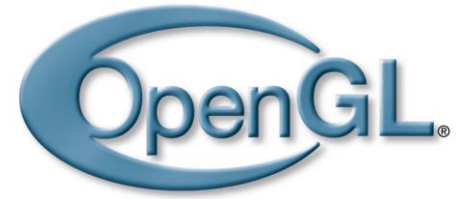
API para desarrollar aplicaciones en dos y tres dimensiones. Compuesto aproximadamente por 250 funciones distintas. Su contraparte es Direct3D desarrollado por Microsoft.

OpenGL - Pygame



- Modulo multi plataforma diseñado para ser usado con Python.
- Incluye librerías de computación gráfica, sonido, etc.
- Permite la creación de GUIs.
- Manejo de ventanas, eventos, sprites, etc.
- Descarga: U-Cursos → Enlaces.
- Permite utilizar la API de OpenGL.

OpenGL - Inicio



- Importación de librerías:

```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
from CC3501Utils import * #Modulo con utilidades del curso
#Vector, coordenadas esféricas, etc.
```

OpenGL – Generación ventana



```
pygame.init() #Inicializa pygame
pygame.display.set_mode((width,height),OPENGL|DOUBLEBUF )
#genera una ventana de (ancho, alto)
#ademas indica que es una ventana renderizada con OPENGL
#y que se utilizara doble buffer
pygame.display.set_caption("nombre de la ventana")
#bajo este código se deben anidar las acciones que se van
#a desarrollar en un loop while para pasar los frame
pygame.quit() #cerrar pygame
```

OpenGL – Loops y eventos

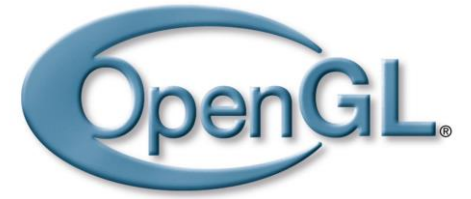


```
run=True #condición booleana True para iterar
#infinitamente hasta quebrar el ciclo con otra condición
while run:

    #para cada evento almacenado en "obtener eventos"
    for event in pygame.event.get():
        if event.type==QUIT: #click sobre cerrar para salir
            run=False #hacer False la condición para terminar el loop
        if event.type==KEYDOWN: #al presionar una tecla
            if event.key==K_ESCAPE: #si se presiona ESC
                run=False #se corta el ciclo

    pygame.display.flip() #redibuja la ventana con el buffer almacenado
    #necesario para actualizar la imagen
    pygame.time.wait(1000/30) #ajusta frecuencia a 30 FPS
```

OpenGL – Gráficos en 2D

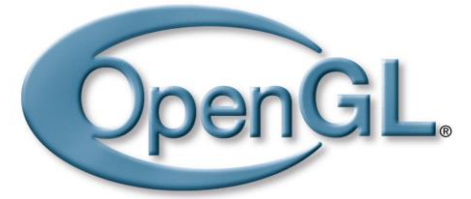


- Los vértices son la unidad básica de trabajo en OpenGL. Cada figura debe ser dibujada especificando previamente la posición de sus vértices.
- Funciones:

```
glVertex2f(x,y) #recibe x,y float  
glVertex2fv(v) #recibe un vector  
glVertex2i(x,y) #recibe x,y int
```

- Forma análoga para 3D.

OpenGL – Primitivas

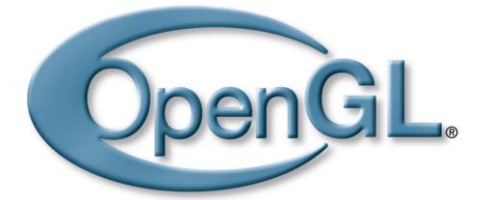


- Los vértices permiten dar una "guía" para posteriormente dibujar las primitivas.

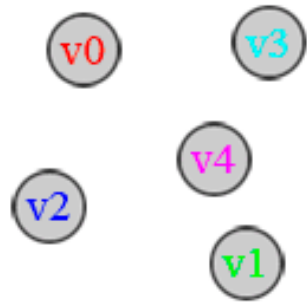
```
glBegin(PRIMITIVA) #comienza a dibujar  
glEnd() #termina de dibujar
```

- Forma análoga para 3D.

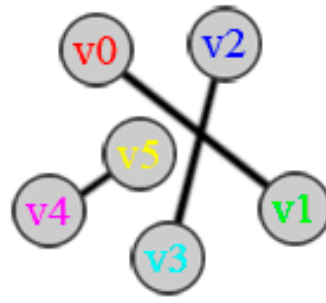
OpenGL – Primitives



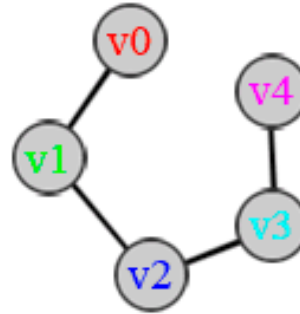
Geometric Primitive Types in OpenTK.OpenGL (defined Clockwise)



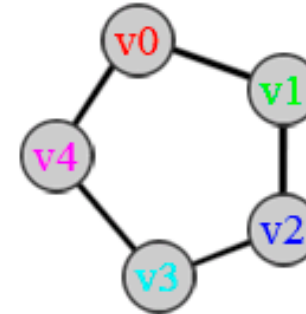
Points



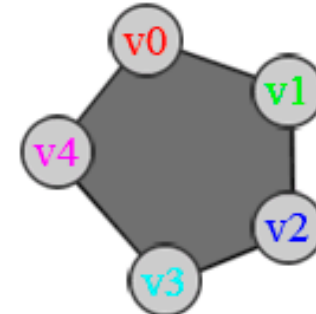
Lines



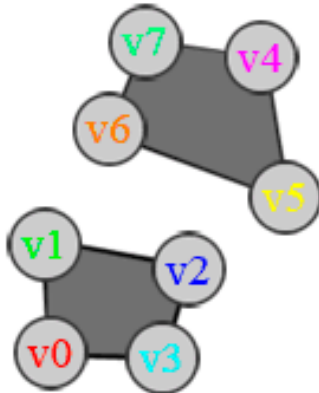
LineStrip



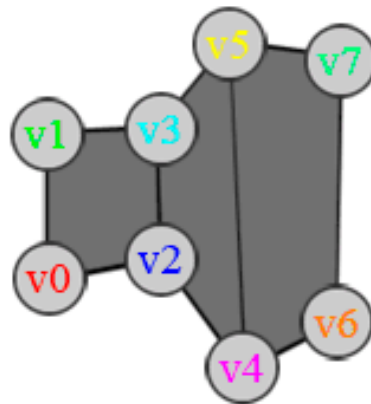
LineLoop



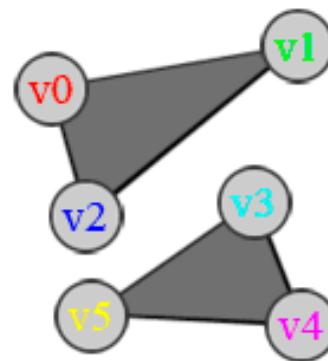
Polygon



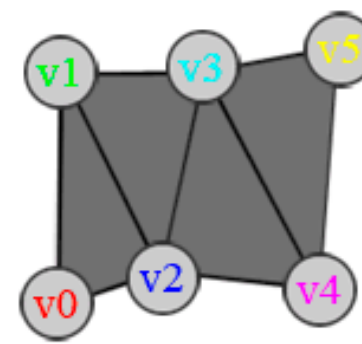
Quads



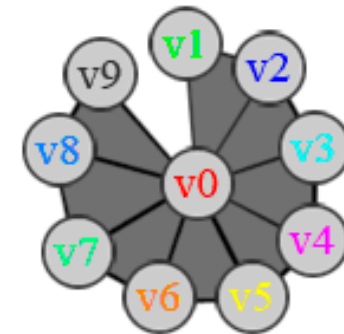
QuadStrip



Triangles

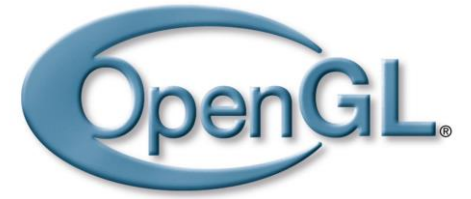


TriangleStrip

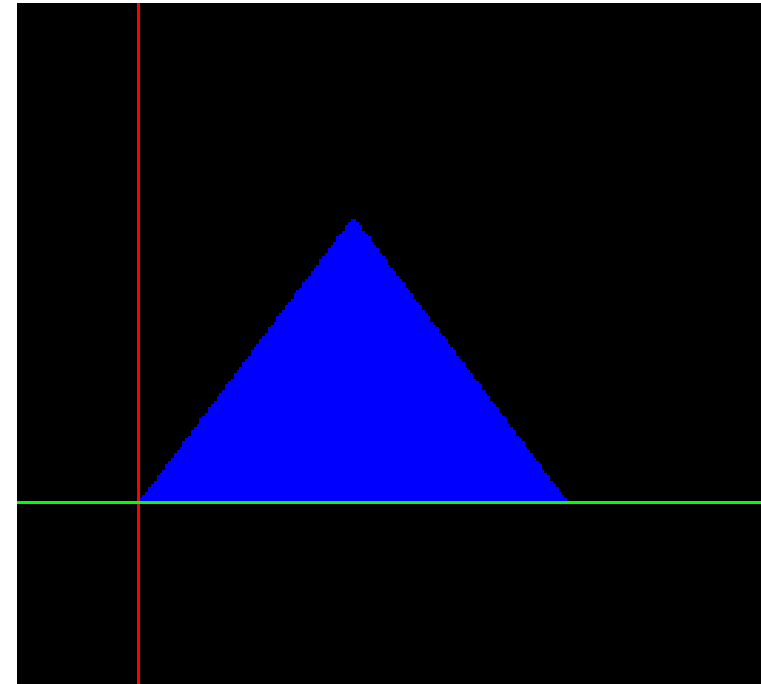


TriangleFan

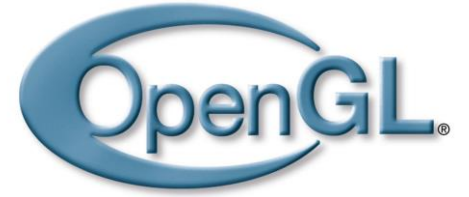
OpenGL – Ejemplo primitivas



```
glBegin(GL_TRIANGLES)
glVertex2f(0.0, 0.0)
glVertex2f(150.0, 0.0)
glVertex2f(75.0, 100.0)
glEnd()
```

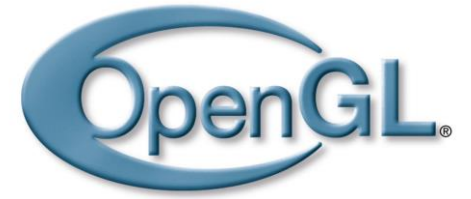


OpenGL – Transformaciones

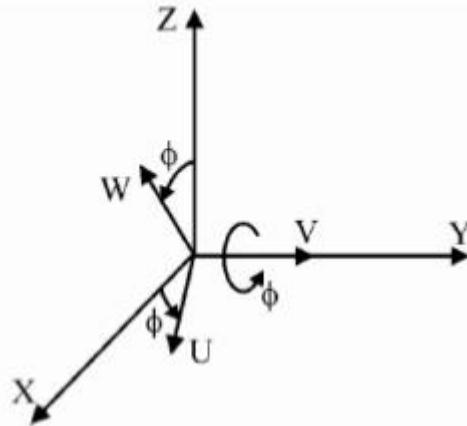


- También es posible aplicar transformaciones geométricas a las figuras (listas de vértices)
 - Rotación
 - Traslación
 - Escalamiento
- OpenGL funciona con un sistema de matrices para todas las operaciones.

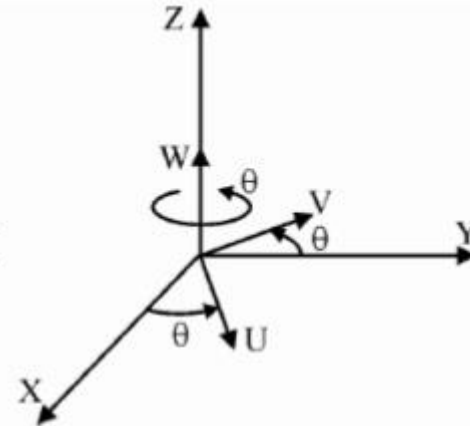
OpenGL – Transformaciones



- ¿Te acuerdas?

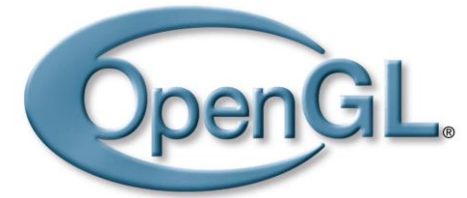


$$\mathbf{R}(y, \phi) = \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{bmatrix}$$



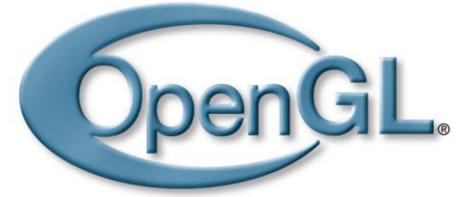
$$\mathbf{R}(z, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

OpenGL – Transformaciones



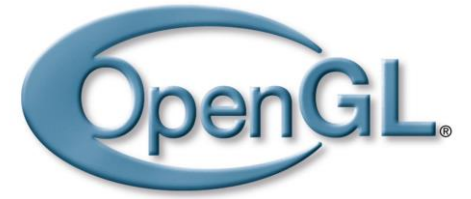
- Para aplicar dichas transformaciones sólo basta usar las funciones de la api de OpenGL:
- Traslación: `glTranslatef(x,y,z)`
- Escalamiento: `glScalef(x,y,z)`
 - Aplica una transformación de escala con los factores x,y,z en cada eje. Valores negativos producen reflexiones en torno al eje.
- Rotación: `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 – Ejemplo



- Ahora queremos dibujar un triángulo en pantalla, con un color en cada vértice, tal como se muestra en la figura siguiente.
- ¿Qué necesitamos?

