CURSO: CC322 - 2017

Practica 5. .

1. Introducción

Objetivo general:
-Conocer Texturas. Mapas procedimentales Mapas UVW. Mapas de textura. Mapas de reflexión. Bump map. Light map. Mip map.

2 . Recursos Informáticos

https://www.opengl.org
https://www.khronos.org/opengl/wiki/Getting_Started
https://www.khronos.org/opengl/wiki/Category:Core_API_Reference
https://www.python.org/
https://www.jetbrains.com/pycharm/

3. DESARROLLO

1. Verifique si tiene instalado Python. Si no está Proceda a instalarlo

```
sudo apt-get install python2.7
type python3 python2.7 python3.5 python2 python
sudo apt-get install python-opengl
```

2. Verifique si tiene instalado pip. Si no está proceda a instalarlo

```
sudo apt-get install python-pip
```

3. Verifique si tiene un editor de texto adecuado para editar programas en Python (p.ej. Geany, SublimeText, u otro con el que esté familiarizado) si no está – Proceda a Instalarlo. Se recomienda PyCharm , version community - https://www.jetbrains.com/pycharm/

3.Verifique si tiene instaladas las librerias OPENGL Y GLUT. Si no están proceda a instalarlas.

```
sudo apt-get install freeglut3-dev
```

4. verifique que tiene instalada las librerias PIL ,PyDispatcher, PyVRML97, OpenGLContext. En caso contrario, proceda a instalarlas

```
sudo apt-get install python-pil
sudo pip install PyDispatcher PyVRML97 OpenGLContext
```

5. Al final entregara un archivo con el nombre CC322_Lab05_<Nombre_apellido>.zip con los archivos generados en la practica.

4 . Ejemplos Prácticos

Ejemplo 4.1.

Copie el programa siguiente o bajelo del sitio web del curso. (Lab05_1.py). Tambien encontrará las imagenes utilizadas.

```
#! /usr/bin/env python
import string
__version__ = string.split('$Revision: 1.1.1.1 $')[1]
__date__ = string.join(string.split('$Date: 2015/02/15 19:25:21 $')[1:3], ' ')
__author__ = 'Modificado por ___________'
#
#
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import sys
from OpenGL.GL.ARB.multitexture import *
from PIL.Image import *
from math import *
# Some api in the chain is translating the keystrokes to this octal string
# so instead of saying: ESCAPE = 27, we use the following.
ESCAPE = '\033'
# Number of the glut window.
window = 0
# Rotations for cube.
rot = 0.0
def LoadTexture(name):
    # global texture
    image = open(name)
    ix = image.size[0]
    iy = image.size[1]
    image = image.tobytes("raw", "RGBX", 0, -1)
    # Create Texture
    id = glGenTextures(1)
    glBindTexture(GL_TEXTURE_2D, id)  # 2d texture (x and y size)
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1)
    glTexImage2D(GL_TEXTURE_2D, 0, 3, ix, iy, 0, GL_RGBA, GL_UNSIGNED_BYTE, image)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)
    return id
# A general OpenGL initialization function.  Sets all of the initial parameters.
def InitGL(Width, Height):  # We call this right after our OpenGL window is created.
    global textures, glMultiTexCoord2f, glActiveTexture, GL_TEXTURE0, GL_TEXTURE1
    print 'Checking for extension support'
    if not glMultiTexCoord2f:
        print 'No OpenGL v1.3 built-in multi-texture support, checking for extension'
        if not glMultiTexCoord2fARB:
            print 'No GL_ARB_multitexture support, sorry, cannot run this demo!'
            sys.exit(1)
        else:
            glMultiTexCoord2f = glMultiTexCoord2fARB
            glActiveTexture = glActiveTextureARB
            GL_TEXTURE0 = GL_TEXTURE0_ARB
            GL_TEXTURE1 = GL_TEXTURE1_ARB
    else:
        print 'Using OpenGL v1.3 built-in multi-texture support'
    try:
        if not glInitMultitextureARB():
            print "Help!  No GL_ARB_multitexture"
            sys.exit(1)
    except NameError, err:
        # don't need to init a built-in (or an extension any more, for that matter)
        pass
    glActiveTexture(GL_TEXTURE0)
    LoadTexture('Wall2.bmp')
    glEnable(GL_TEXTURE_2D)
    glActiveTexture(GL_TEXTURE1)
    LoadTexture('NeHe.bmp')
    glEnable(GL_TEXTURE_2D)
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_BLEND)
    glClearColor(0.0, 0.0, 0.0, 0.0)  # This Will Clear The Background Color To Black
    glClearDepth(1.0)  # Enables Clearing Of The Depth Buffer
    glDepthFunc(GL_LESS)  # The Type Of Depth Test To Do
    glEnable(GL_DEPTH_TEST)  # Enables Depth Testing
    glShadeModel(GL_SMOOTH)  # Enables Smooth Color Shading
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()  # Reset The Projection Matrix
    # Calculate The Aspect Ratio Of The Window
    gluPerspective(45.0, float(Width) / float(Height), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
# The function called when our window is resized (which shouldn't happen if you enable fullscreen, below)
```

```python
def ReSizeGLScene(Width, Height):
    if Height == 0:  # Prevent A Divide By Zero If The Window Is Too Small
        Height = 1
    glViewport(0, 0, Width, Height)  # Reset The Current Viewport And Perspective Transformation
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(45.0, float(Width) / float(Height), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
deg_rad = pi / 180.0
# The main drawing function.
def DrawGLScene():
    global rot, texture
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)  # Clear The Screen And The Depth Buffer
    glLoadIdentity()  # Reset The View
    glTranslatef(0.0, 0.0, -5.0)  # Move Into The Screen
    glRotatef(rot, 1.0, 0.0, 0.0)  # Rotate The Cube On It's X Axis
    glRotatef(rot, 0.0, 1.0, 0.0)  # Rotate The Cube On It's Y Axis
    glRotatef(rot, 0.0, 0.0, 1.0)  # Rotate The Cube On It's Z Axis
    # Note there does not seem to be support for this call.
    # glBindTexture(GL_TEXTURE_2D,texture) # Rotate The Pyramid On It's Y Axis
    p = cos(rot * deg_rad) ** 2
    glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, (p, p, p, 1))
    glBegin(GL_QUADS)  # Start Drawing The Cube
    # Front Face (note that the texture's corners have to match the quad's corners)
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(-1.0, -1.0, 1.0)  # Bottom Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(1.0, -1.0, 1.0)  # Bottom Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(1.0, 1.0, 1.0)  # Top Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(-1.0, 1.0, 1.0)  # Top Left Of The Texture and Quad
    # Back Face
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(-1.0, -1.0, -1.0)  # Bottom Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(-1.0, 1.0, -1.0)  # Top Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(1.0, 1.0, -1.0)  # Top Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(1.0, -1.0, -1.0)  # Bottom Left Of The Texture and Quad
    # Top Face
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(-1.0, 1.0, -1.0)  # Top Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(-1.0, 1.0, 1.0)  # Bottom Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(1.0, 1.0, 1.0)  # Bottom Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(1.0, 1.0, -1.0)  # Top Right Of The Texture and Quad
    # Bottom Face
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(-1.0, -1.0, -1.0)  # Top Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(1.0, -1.0, -1.0)  # Top Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(1.0, -1.0, 1.0)  # Bottom Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(-1.0, -1.0, 1.0)  # Bottom Right Of The Texture and Quad
    # Right face
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(1.0, -1.0, -1.0)  # Bottom Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(1.0, 1.0, -1.0)  # Top Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(1.0, 1.0, 1.0)  # Top Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(1.0, -1.0, 1.0)  # Bottom Left Of The Texture and Quad
    # Left Face
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 0.0)
    glVertex3f(-1.0, -1.0, -1.0)  # Bottom Left Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 0.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 0.0)
    glVertex3f(-1.0, -1.0, 1.0)  # Bottom Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 1.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 1.0, 1.0)
    glVertex3f(-1.0, 1.0, 1.0)  # Top Right Of The Texture and Quad
    glMultiTexCoord2f(GL_TEXTURE0_ARB, 0.0, 1.0)
    glMultiTexCoord2f(GL_TEXTURE1_ARB, 0.0, 1.0)
    glVertex3f(-1.0, 1.0, -1.0)  # Top Left Of The Texture and Quad
    glEnd()   # Done Drawing The Cube
    rot = (rot + 0.2) % 360  # rotation
    #  since this is double buffered, swap the buffers to display what just got drawn.
    glutSwapBuffers()
# The function called whenever a key is pressed. Note the use of Python tuples to pass in: (key, x, y)
def keyPressed(*args):
    # If escape is pressed, kill everything.
    if args[0] == ESCAPE:
        sys.exit()
def main():
    global window
    glutInit(sys.argv)
    # Select type of Display mode:
    #  Double buffer
    #  RGBA color
    # Alpha components supported
    # Depth buffer
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
    # get a 640 x 480 window
    glutInitWindowSize(640, 480)
    # the window starts at the upper left corner of the screen
    glutInitWindowPosition(0, 0)
    # Okay, like the C version we retain the window id to use when closing, but for those of you new
    # to Python (like myself), remember this assignment would make the variable local and not global
    # if it weren't for the global declaration at the start of main.
    window = glutCreateWindow("Lab. Semana 05: texturas ")
    # Register the drawing function with glut, BUT in Python land, at least using PyOpenGL, we need to
    # set the function pointer and invoke a function to actually register the callback, otherwise it
    # would be very much like the C version of the code.
    glutDisplayFunc(DrawGLScene)
    # Uncomment this line to get full screen.
    # glutFullScreen()
    # When we are doing nothing, redraw the scene.
    glutIdleFunc(DrawGLScene)
    # Register the function called when our window is resized.
    glutReshapeFunc(ReSizeGLScene)
    # Register the function called when the keyboard is pressed.
    glutKeyboardFunc(keyPressed)
    # Initialize our window.
    InitGL(640, 480)
    # Start Event Processing Engine
    glutMainLoop()
# Print message to console, and kick off the main to get it rolling.
if __name__ == "__main__":
    print "Hit ESC key to quit."
    main()
```

a) Modifique el programa de tal manera que el 3 caras del cubo tengan una secuencia y otras 3 caras tengan una figura fija. Utilice imágenes diferentes. Grabe el archivo con el nombre lab05_11.py

Ejemplo 4.2.

Copie el programa siguiente o bajelo del sitio web del curso. (Lab05_2.py). Tambien encontrará las imagenes utilizadas.

```
##! /usr/bin/env python
import string
__version__ = string.split('$Revision: 1.1.1.1 $')[1]
__date__ = string.join(string.split('$Date: 2007/02/15 19:25:20 $')[1:3], ' ')
__author__ = 'Modificado por ___________'
#
#
from OpenGL.GL import *
```

```python
from OpenGL.GLUT import *
from OpenGL.GLU import *
import sys
from PIL.Image import *
# Some api in the chain is translating the keystrokes to this octal string
# so instead of saying: ESCAPE = 27, we use the following.
ESCAPE = '\033'
# Number of the glut window.
window = 0
# Rotations for cube.
xrot = yrot = zrot = 0.0
texture_num = 2
object = 0
light = 0
def LoadTextures():
    global texture_num, textures
    image = open("Terrain.bmp")
    ix = image.size[0]
    iy = image.size[1]
    #image = image.tostring("raw", "RGBX", 0, -1)
    image = image.tobytes("raw", "RGBX", 0, -1)
    # Create Texture
    textures = glGenTextures(3)
    glBindTexture(GL_TEXTURE_2D, int(textures[0]))   # 2d texture (x and y size)
    glPixelStorei(GL_UNPACK_ALIGNMENT,1)
    glTexImage2D(GL_TEXTURE_2D, 0, 3, ix, iy, 0, GL_RGBA, GL_UNSIGNED_BYTE, image)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)
    # Create Linear Filtered Texture
    glBindTexture(GL_TEXTURE_2D, int(textures[1]))
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR)
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR)
    glTexImage2D(GL_TEXTURE_2D, 0, 3, ix, iy, 0, GL_RGBA, GL_UNSIGNED_BYTE, image)
    # Create MipMapped Texture
    glBindTexture(GL_TEXTURE_2D, int(textures[2]))
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR)
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR_MIPMAP_NEAREST)
    gluBuild2DMipmaps(GL_TEXTURE_2D, 3, ix, iy, GL_RGBA, GL_UNSIGNED_BYTE, image)
# A general OpenGL initialization function.  Sets all of the initial parameters.
def InitGL(Width, Height):          # We call this right after our OpenGL window is created.
    global quadratic
    LoadTextures()
    quadratic = gluNewQuadric()
    gluQuadricNormals(quadratic, GLU_SMOOTH)      # Create Smooth Normals (NEW)
    gluQuadricTexture(quadratic, GL_TRUE)        # Create Texture Coords (NEW)
    glEnable(GL_TEXTURE_2D)
    glClearColor(0.0, 0.0, 0.0, 0.0)   # This Will Clear The Background Color To Black
    glClearDepth(1.0)              # Enables Clearing Of The Depth Buffer
    glDepthFunc(GL_LESS)            # The Type Of Depth Test To Do
    glEnable(GL_DEPTH_TEST)             # Enables Depth Testing
    glShadeModel(GL_SMOOTH)             # Enables Smooth Color Shading
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()               # Reset The Projection Matrix
                                        # Calculate The Aspect Ratio Of The Window
    gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
    glLightfv(GL_LIGHT0, GL_AMBIENT, (0.5, 0.5, 0.5, 1.0))    # Setup The Ambient Light
    glLightfv(GL_LIGHT0, GL_DIFFUSE, (1.0, 1.0, 1.0, 1.0))    # Setup The Diffuse Light
    glLightfv(GL_LIGHT0, GL_POSITION, (0.0, 0.0, 2.0, 1.0))    # Position The Light
    glEnable(GL_LIGHT0)                # Enable Light One
# The function called when our window is resized (which shouldn't happen if you enable fullscreen, below)
def ReSizeGLScene(Width, Height):
    if Height == 0:                   # Prevent A Divide By Zero If The Window Is Too Small
        Height = 1
    glViewport(0, 0, Width, Height)       # Reset The Current Viewport And Perspective Transformation
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
def DrawCube():
    glBegin(GL_QUADS)           # Start Drawing The Cube
    # Front Face (note that the texture's corners have to match the quad's corners)
    glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)   # Bottom Left Of The Texture and Quad
    glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)   # Bottom Right Of The Texture and Quad
    glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0,  1.0)   # Top Right Of The Texture and Quad
    glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0,  1.0)   # Top Left Of The Texture and Quad
    # Back Face
    glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)   # Bottom Right Of The Texture and Quad
    glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)   # Top Right Of The Texture and Quad
    glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)   # Top Left Of The Texture and Quad
```

```python
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)   # Bottom Left Of The Texture and Quad
        # Top Face
        glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)   # Top Left Of The Texture and Quad
        glTexCoord2f(0.0, 0.0); glVertex3f(-1.0,  1.0,  1.0)   # Bottom Left Of The Texture and Quad
        glTexCoord2f(1.0, 0.0); glVertex3f( 1.0,  1.0,  1.0)   # Bottom Right Of The Texture and Quad
        glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)   # Top Right Of The Texture and Quad
        # Bottom Face
        glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, -1.0, -1.0)   # Top Right Of The Texture and Quad
        glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, -1.0, -1.0)   # Top Left Of The Texture and Quad
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)   # Bottom Left Of The Texture and Quad
        glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)   # Bottom Right Of The Texture and Quad
        # Right face
        glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)   # Bottom Right Of The Texture and Quad
        glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)   # Top Right Of The Texture and Quad
        glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0,  1.0)   # Top Left Of The Texture and Quad
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)   # Bottom Left Of The Texture and Quad
        # Left Face
        glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)   # Bottom Left Of The Texture and Quad
        glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)   # Bottom Right Of The Texture and Quad
        glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0,  1.0)   # Top Right Of The Texture and Quad
        glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)   # Top Left Of The Texture and Quad
        glEnd();            # Done Drawing The Cube
    # The main drawing function.
    def DrawGLScene():
        global xrot, yrot, zrot, textures, texture_num, object, quadratic, light
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) # Clear The Screen And The Depth Buffer
        glLoadIdentity()                # Reset The View
        glTranslatef(0.0,0.0,-5.0)        # Move Into The Screen
        glRotatef(xrot,1.0,0.0,0.0)           # Rotate The Cube On It's X Axis
        glRotatef(yrot,0.0,1.0,0.0)           # Rotate The Cube On It's Y Axis
        glRotatef(zrot,0.0,0.0,1.0)           # Rotate The Cube On It's Z Axis
        glBindTexture(GL_TEXTURE_2D, int(textures[texture_num]))
        if object == 0:
            DrawCube()
        elif object == 1:
            glTranslatef(0.0,0.0,-1.5)        # Center The Cylinder
            gluCylinder(quadratic,1.0,1.0,3.0,32,32)   # A Cylinder With A Radius Of 0.5 And A Height Of 2
        xrot  = xrot + 0.2          # X rotation
        yrot = yrot + 0.2            # Y rotation
        zrot = zrot + 0.2            # Z rotation
        #  since this is double buffered, swap the buffers to display what just got drawn.
        glutSwapBuffers()
    # The function called whenever a key is pressed
    def keyPressed(key, x, y):
        global object, texture_num, light
        # If escape is pressed, kill everything.
        key = string.upper(key)
        if key == ESCAPE:
            sys.exit()
        elif key == 'L':
            light = not light
        elif key == 'T': #  switch the texture
            texture_num = (texture_num + 1) % 3
        elif key == 'O': #  switch the object
            object = (object + 1) % 2
    def main():
        usage = """Press L to toggle Lighting
    Press T to change textures
    Press O to change objects"""
        print usage
        global window
        glutInit(sys.argv)
        # Select type of Display mode:
        #  Double buffer
        #  RGBA color
        # Alpha components supported
        # Depth buffer
        glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
        # get a 640 x 480 window
        glutInitWindowSize(640, 480)
        # the window starts at the upper left corner of the screen
        glutInitWindowPosition(0, 0)
        # Okay, like the C version we retain the window id to use when closing, but for those of you new
        # to Python (like myself), remember this assignment would make the variable local and not global
        # if it weren't for the global declaration at the start of main.
        window = glutCreateWindow("Lab. Semana 05: texturas ")
        # Register the drawing function with glut, BUT in Python land, at least using PyOpenGL, we need to
        # set the function pointer and invoke a function to actually register the callback, otherwise it
        # would be very much like the C version of the code.
        glutDisplayFunc(DrawGLScene)
        # Uncomment this line to get full screen.
        # glutFullScreen()
        # When we are doing nothing, redraw the scene.
        glutIdleFunc(DrawGLScene)
        # Register the function called when our window is resized.
        glutReshapeFunc(ReSizeGLScene)
        # Register the function called when the keyboard is pressed.
        glutKeyboardFunc(keyPressed)
        # Initialize our window.
        InitGL(640, 480)
        # Start Event Processing Engine
        glutMainLoop()
    # Print message to console, and kick off the main to get it rolling.
    print "Hit ESC key to quit."
    main()
```

a) Modifique el programa de tal manera que presionando repetidamente una tecla se puedan llegar a ver 6 figuras geometricas diferentes (Pueden ser primitivas adicionales), con las texturas y rotacion presentadas en el ejemplo original. Grabe el archivo con el nombre lab05_21.py

b) En un archivo denominado funciones.odt describa las todas las funciones y parametros utilizados que esten relacionados con el mapeo de texturas en este programa. Puede utilizar fuentes de consulta disponibles.