Texturas y Controles

Modelación y Computación Gráfica para Ingenieros

Texturas

- Texture shaders (en easy_shaders.py)
- Wrap mode
- Filter mode

Texturas

- Texture shaders (en easy_shaders.py)
- Wrap mode
- Filter mode

SimpleTextureTransformShaderProgram

- Shader para trabajar texturas en 2D (figuras geométricas)

SimpleTextureModelViewProjectionShaderProgram

- Shader para trabajar texturas en 3D (cuerpos geométricos)

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

textureSimpleSetup

- Función usada para setear texturas

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

imgName

- Parametro que indica el path a la imagen que será usada como textura

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

sWrapMode

- Parametro que indica el modo en que se envolverá la textura en donde será colocada cuando se exceda el rango (0,1) en el eje x.
- Aclaracion: Si se coloca una textura sobre un cuadrado de lado 2 la textura comenzará en 0 en x y terminará en 1 también en x, una vez pasado el 1 en este eje se le debe indicar a la textura como debe completar el espacio que queda, eso se hace mediante este parámetro.
- Algunos posibles valores:
  - GL_REPEAT
  - GL_MIRRORED_REPEAT
  - GL_CLAMP_TO_EDGE
  - GL_MIRROR_CLAMP_TO_EDGE

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

tWrapMode

- Parametro que indica el modo en que se envolverá la textura en donde será colocada cuando se exceda el rango (0,1) en el eje y.
- Aclaracion: Si se coloca una textura sobre un cuadrado de lado 2 la textura comenzará en 0 en y y terminará en 1 también en y, una vez pasado el 1 en este eje se le debe indicar a la textura como debe completar el espacio que queda, eso se hace mediante este parámetro.
- Algunos posibles valores:
  - GL_REPEAT
  - GL_MIRRORED_REPEAT
  - GL_CLAMP_TO_EDGE
  - GL_MIRROR_CLAMP_TO_EDGE

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

minFilterMode

- Parametro que define la función que se usará cuando el pixel que esté siendo texturado se mapee a un área mayor que un elemento de textura.
- Algunos posibles valores:
  - GL_NEAREST
  - GL_LINEAR

```
textureSimpleSetup(imgName, sWrapMode, tWrapMode, minFilterMode, maxFilterMode)
```

maxFilterMode

- Parametro que define la función que se usará cuando el pixel que esté siendo texturado se mapee a un área menor o igual que un elemento de textura.
- Algunos posibles valores:
  - GL_NEAREST
  - GL_LINEAR

Texturas

- Texture shaders (en easy_shaders.py)
- Wrap mode
- Filter mode

Wrap mode

GL_REPEAT

GL_MIRRORED_REPEAT

GL_CLAMP_TO_EDGE

GL_CLAMP_TO_BORDER

Wrap mode

GL_REPEAT

GL_MIRRORED_REPEAT

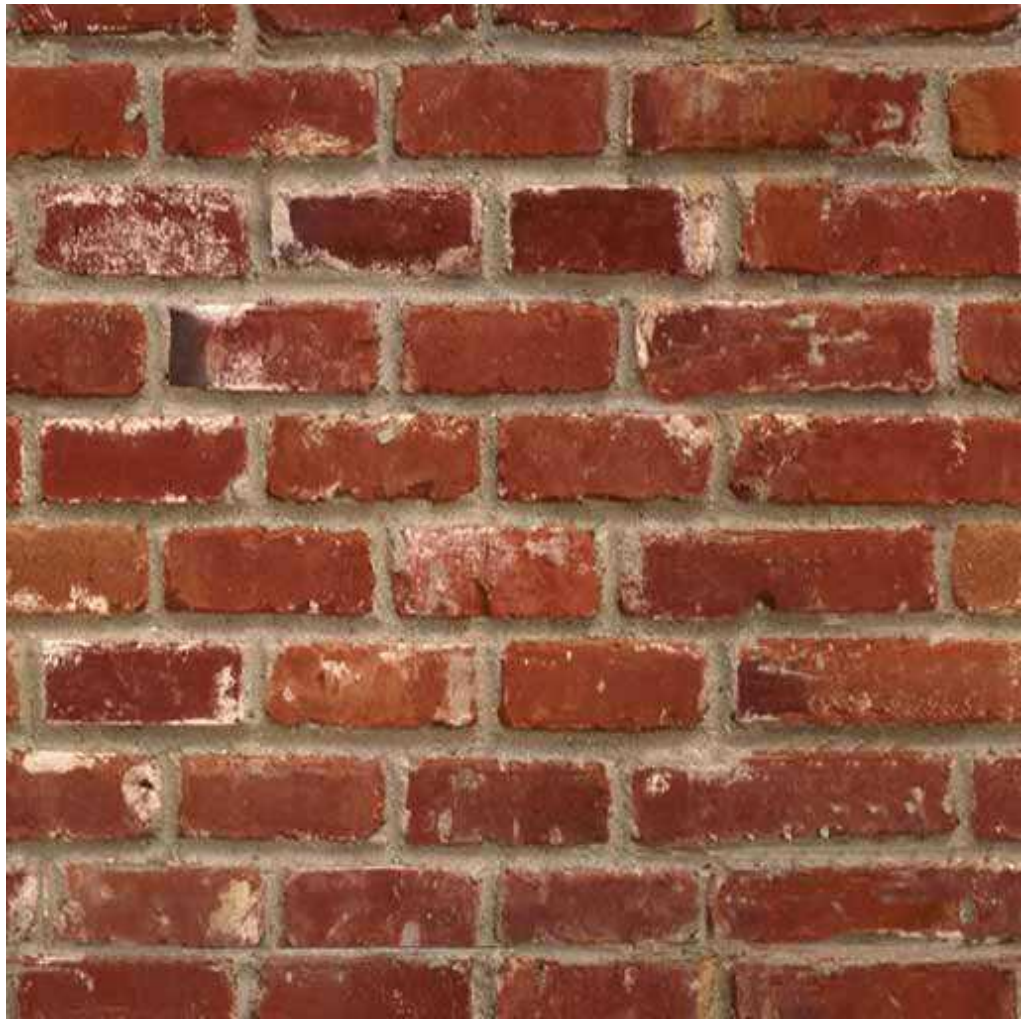
GL_CLAMP_TO_EDGE

GL_CLAMP_TO_BORDER

`sWrapMode`
(eje x)

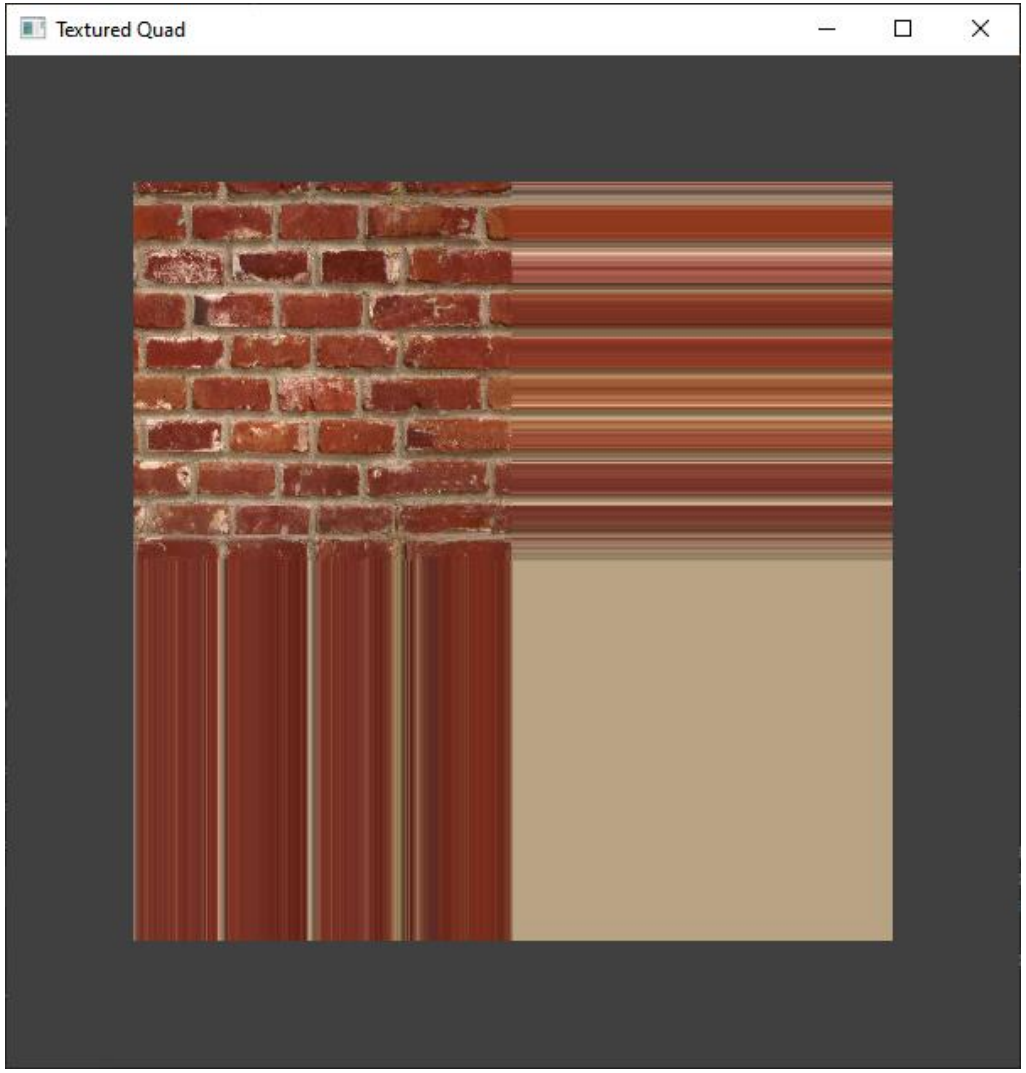
`tWrapMode`
(eje y)

```
imgName= getAssetPath("bricks.jpg")
```



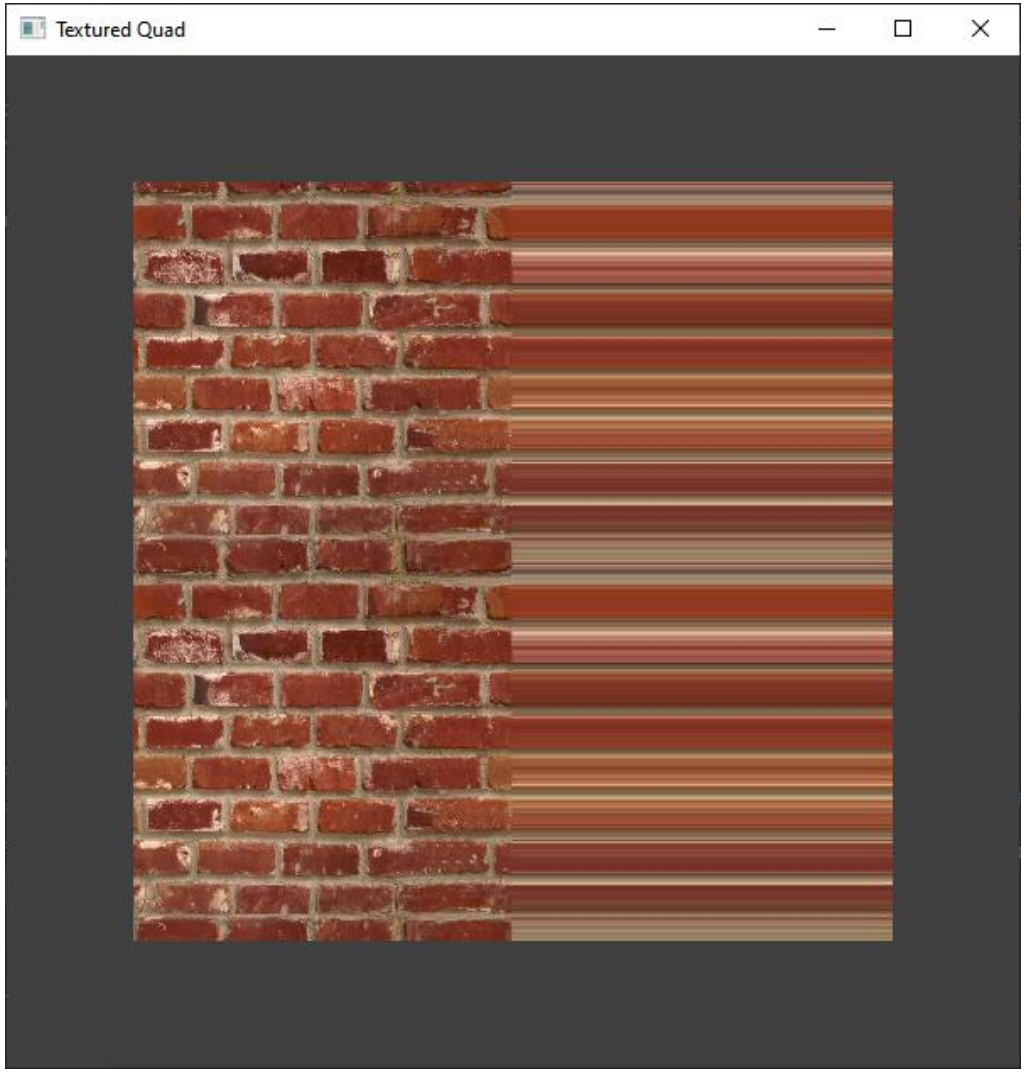
sWrapMode=
GL_CLAMP_TO_EDGE

tWrapMode=
GL_CLAMP_TO_EDGE



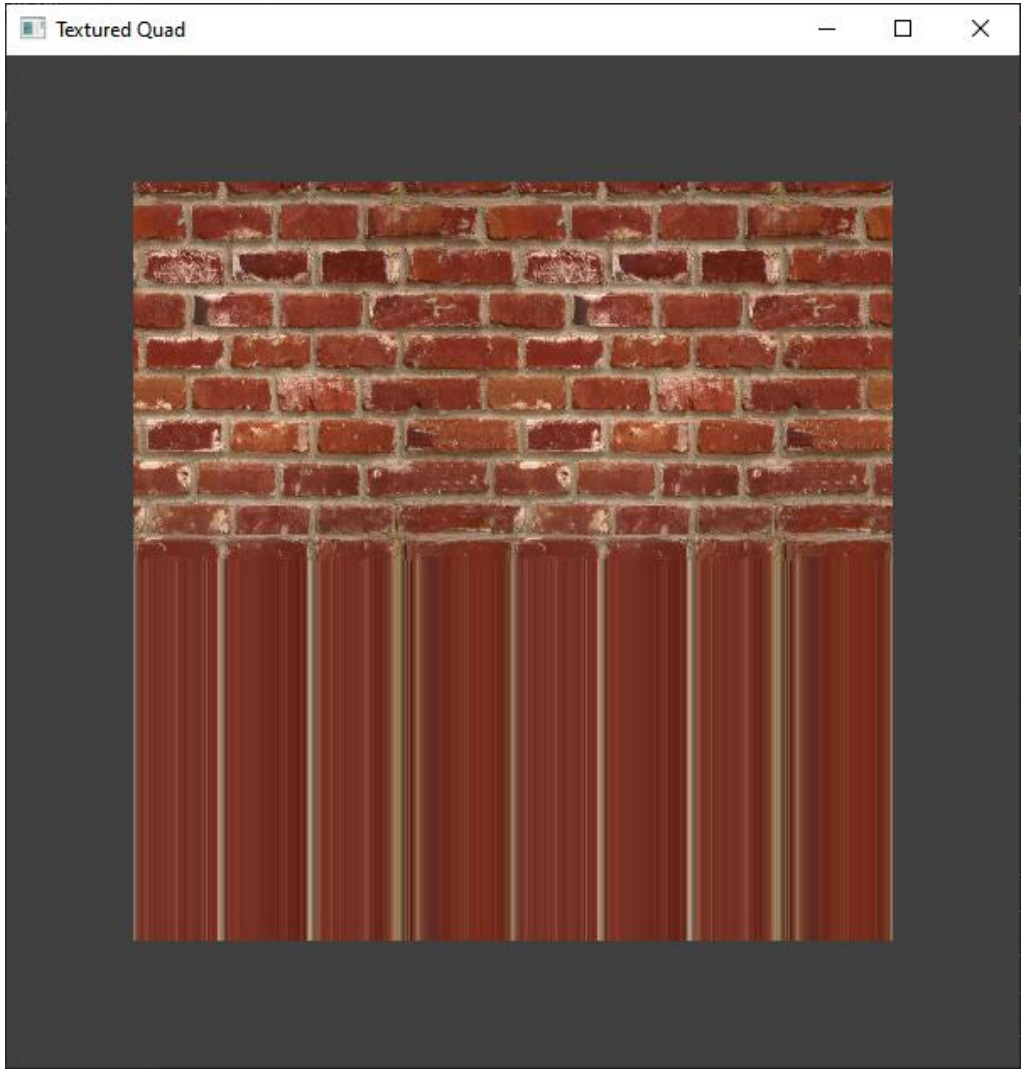
sWrapMode=
    GL_CLAMP_TO_EDGE

tWrapMode=
    GL_REPEAT



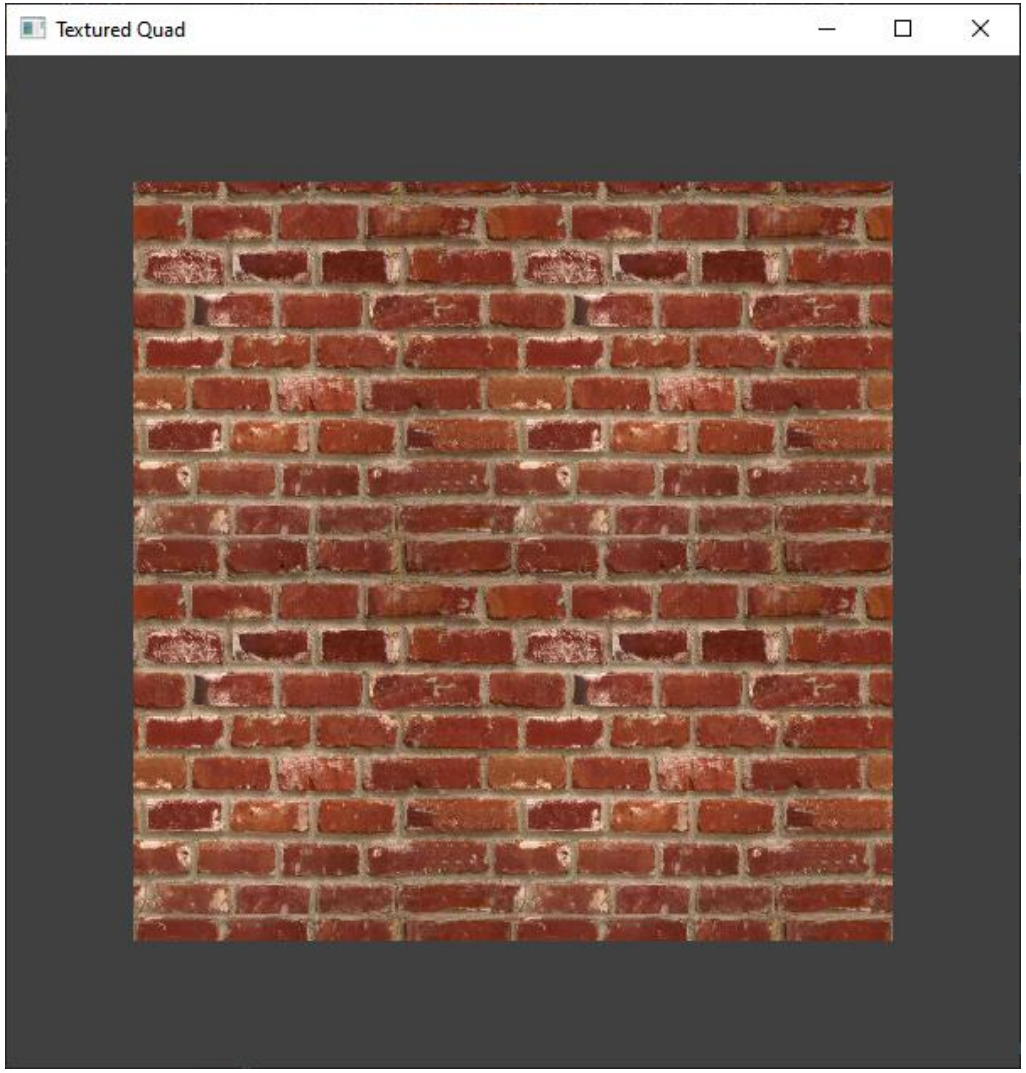
sWrapMode=
    GL_REPEAT

tWrapMode=
    GL_CLAMP_TO_EDGE

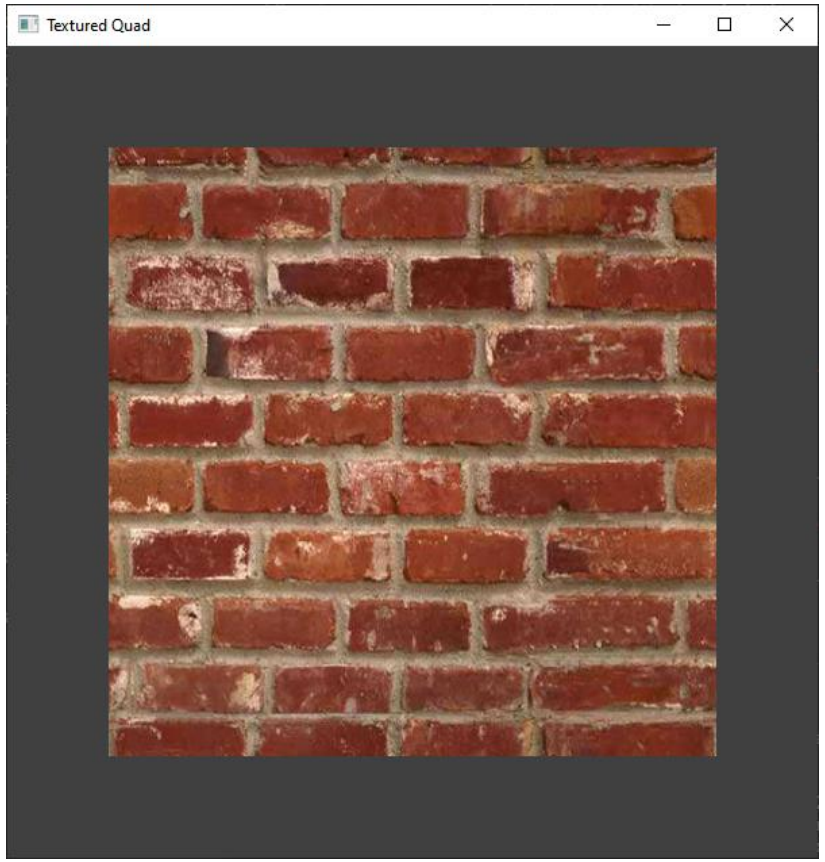


sWrapMode=
GL_REPEAT

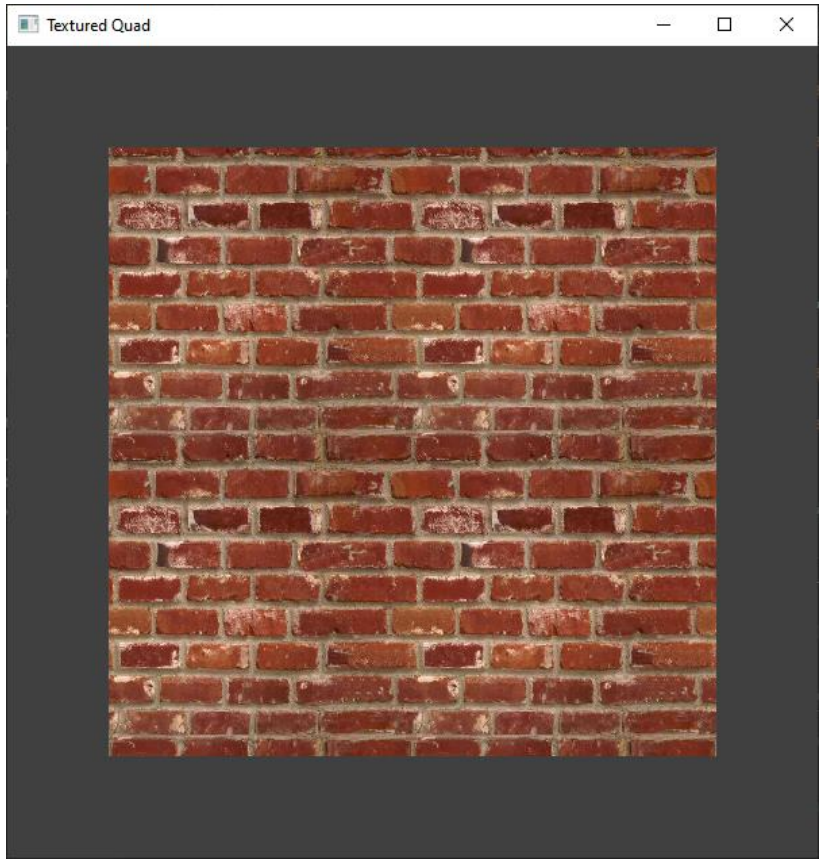
tWrapMode=
GL_REPEAT



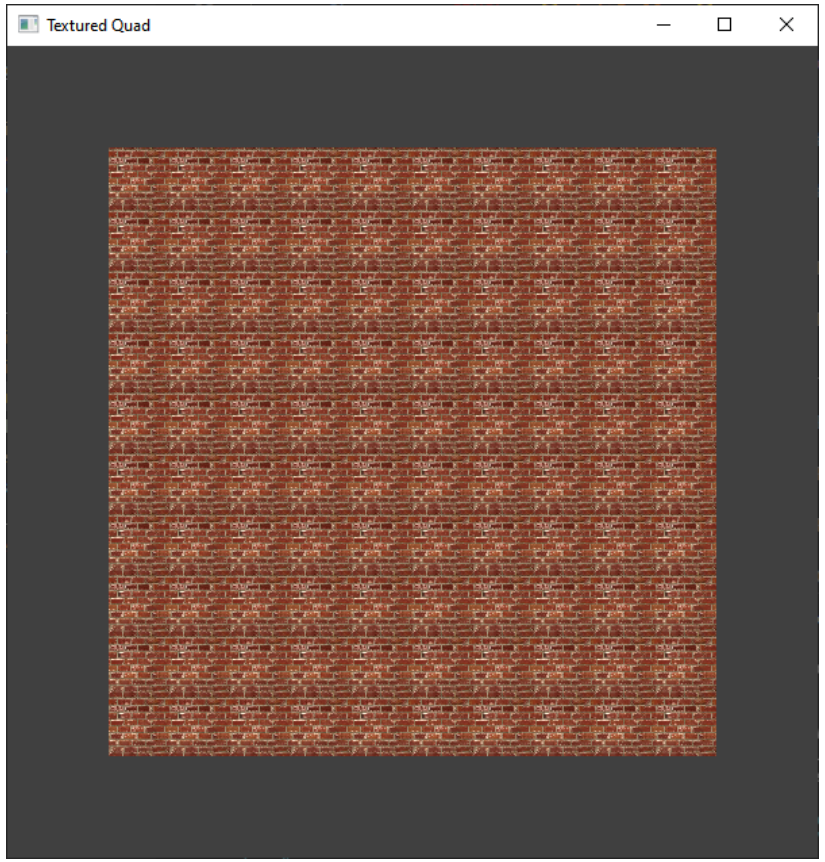
```
shape = bs.createTextureQuad(1, 1)
gpuShape = es.GPUShape().initBuffers()
pipeline.setupVAO(gpuShape)
gpuShape.fillBuffers(shape.vertices, shape.indices, GL_STATIC_DRAW)
gpuShape.texture = es.textureSimpleSetup(
    getAssetPath("bricks.jpg"), GL_REPEAT, GL_REPEAT, GL_LINEAR, GL_LINEAR)
```



```
shape = bs.createTextureQuad(2, 2)
gpuShape = es.GPUShape().initBuffers()
pipeline.setupVAO(gpuShape)
gpuShape.fillBuffers(shape.vertices, shape.indices, GL_STATIC_DRAW)
gpuShape.texture = es.textureSimpleSetup(
    getAssetPath("bricks.jpg"), GL_REPEAT, GL_REPEAT, GL_LINEAR, GL_LINEAR)
```



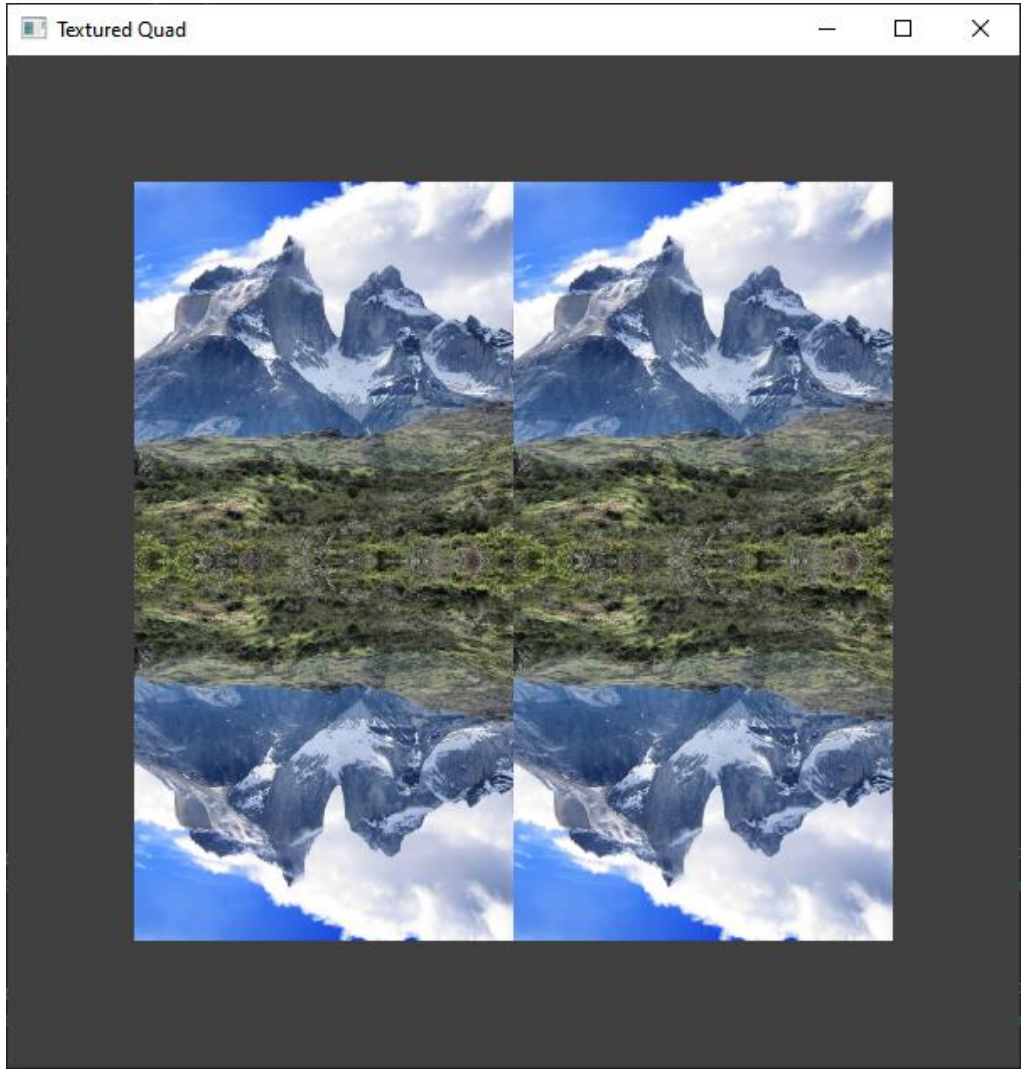
```
shape = bs.createTextureQuad(10, 10)
gpuShape = es.GPUShape().initBuffers()
pipeline.setupVAO(gpuShape)
gpuShape.fillBuffers(shape.vertices, shape.indices, GL_STATIC_DRAW)
gpuShape.texture = es.textureSimpleSetup(
    getAssetPath("bricks.jpg"), GL_REPEAT, GL_REPEAT, GL_LINEAR, GL_LINEAR)
```



```
imgName=
    getAssetPath("torres-del-paine-sq.jpg")
```

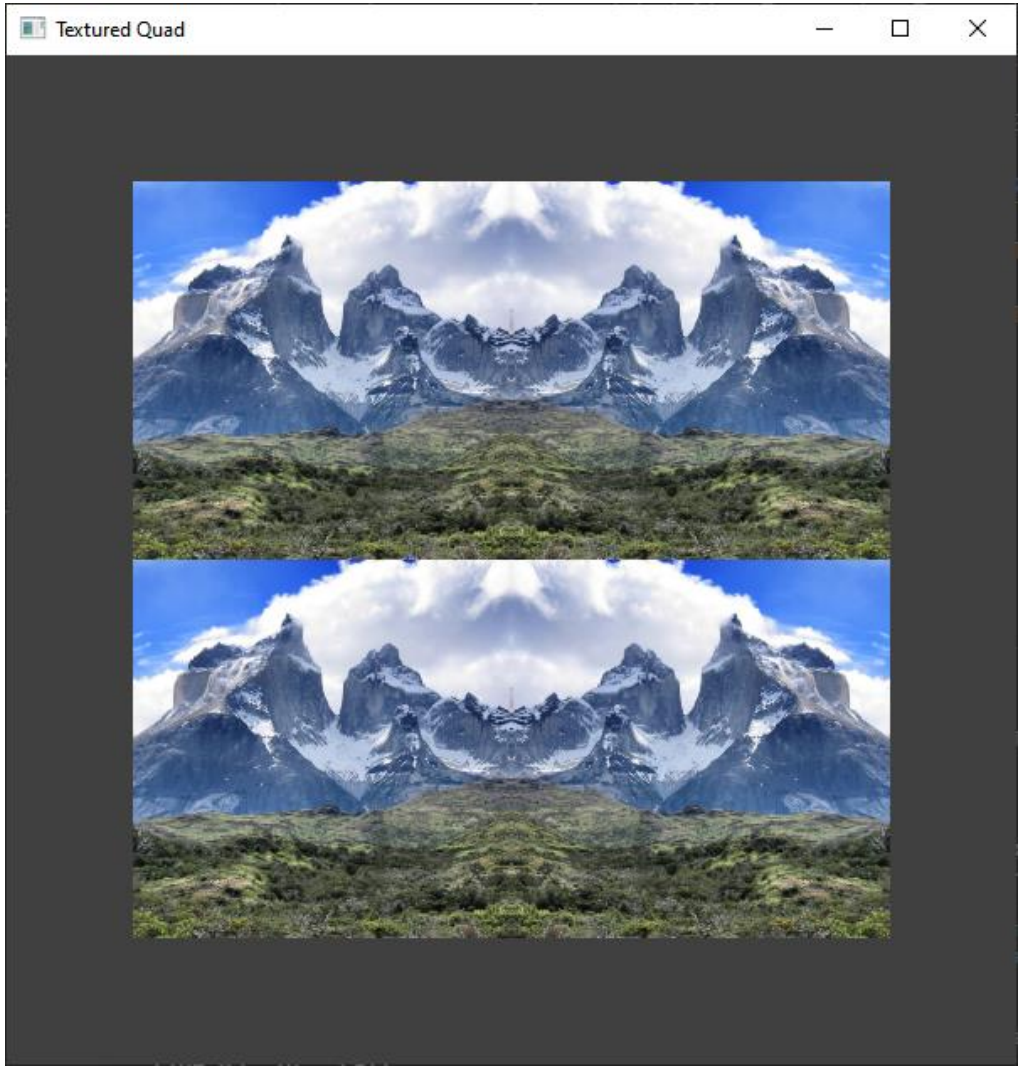
sWrapMode=
    GL_REPEAT

tWrapMode=
    GL_MIRRORED_REPEAT



sWrapMode=
    GL_MIRRORED_REPEAT

tWrapMode=
    GL_REPEAT



Texturas

- Texture shaders (en easy_shaders.py)
- Wrap mode
- Filter mode

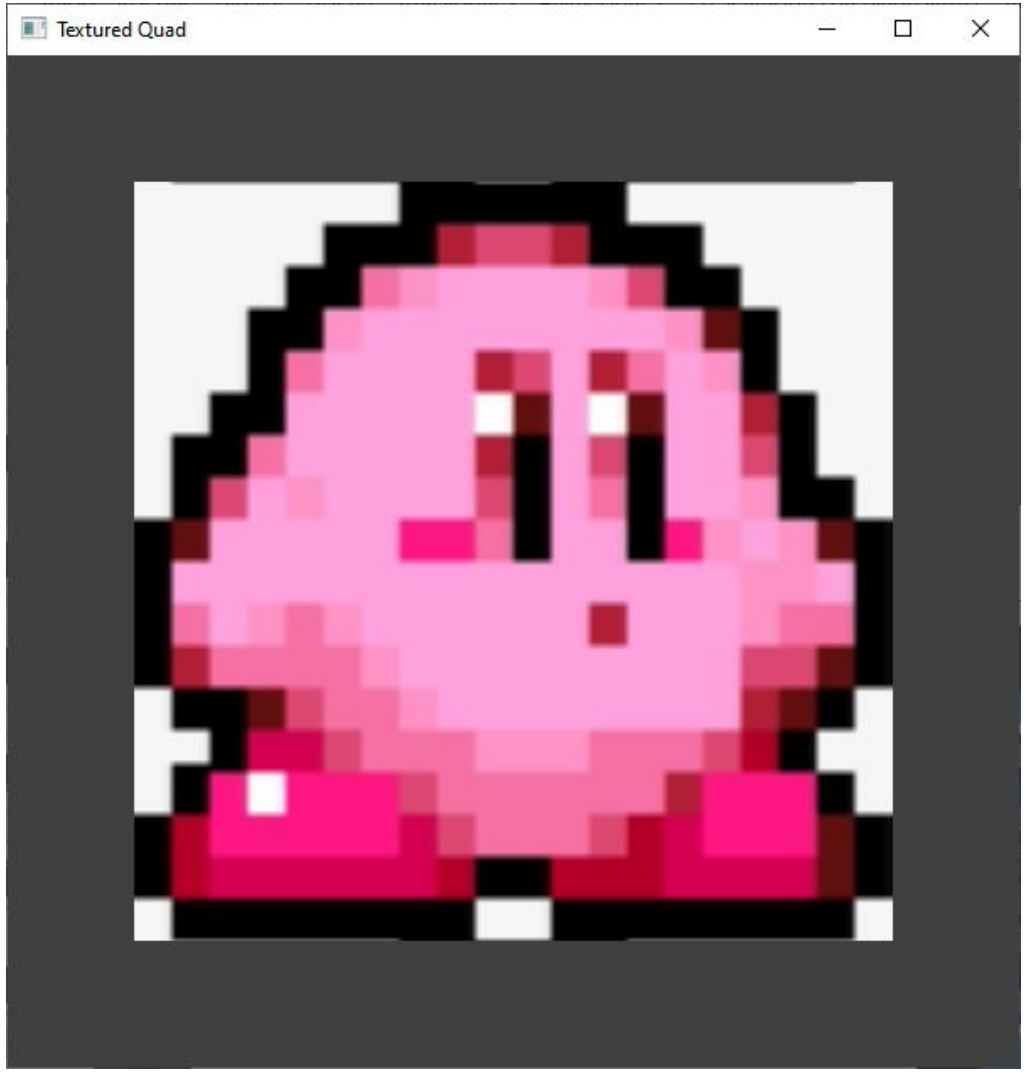
Filter mode

GL_NEAREST

GL_LINEAR

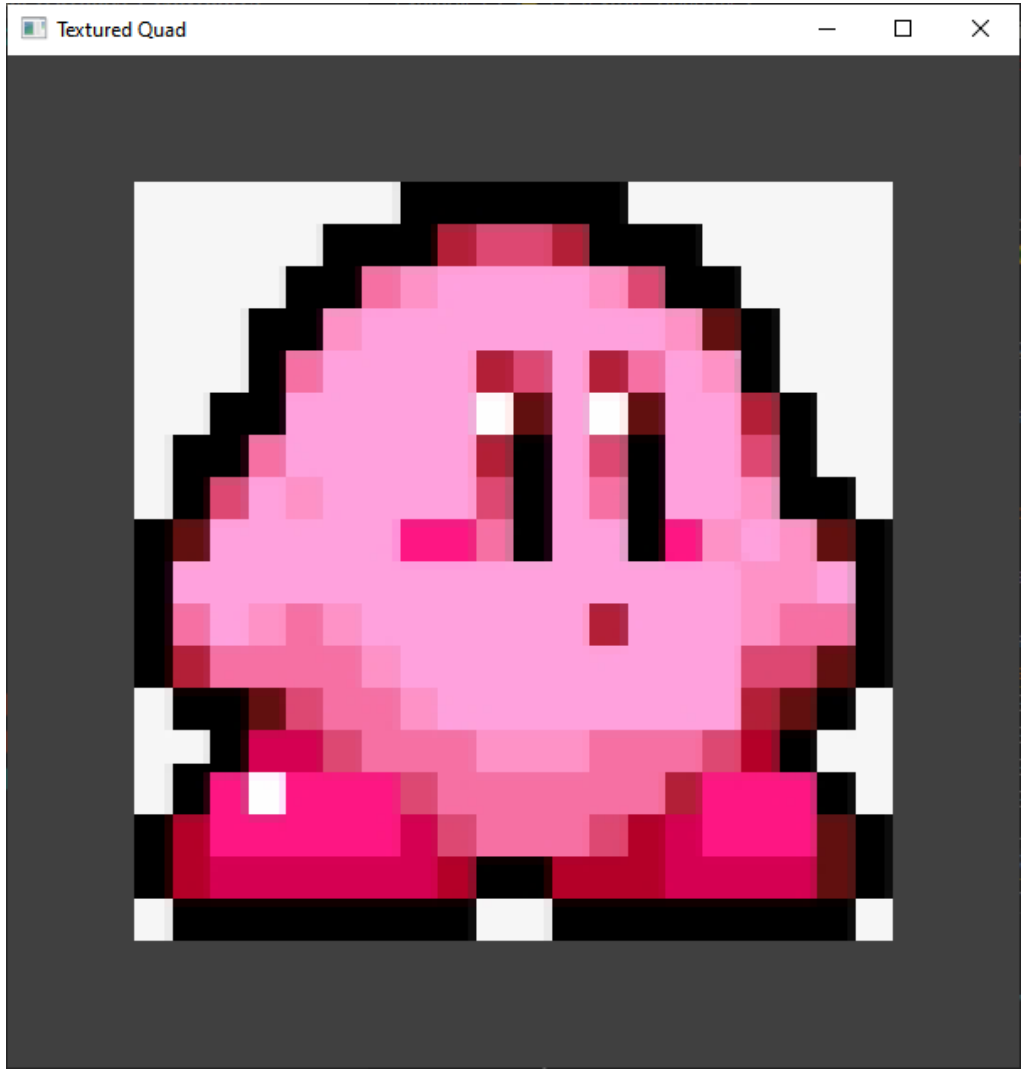
minFilterMode=
GL_LINEAR

maxFilterMode=
GL_LINEAR



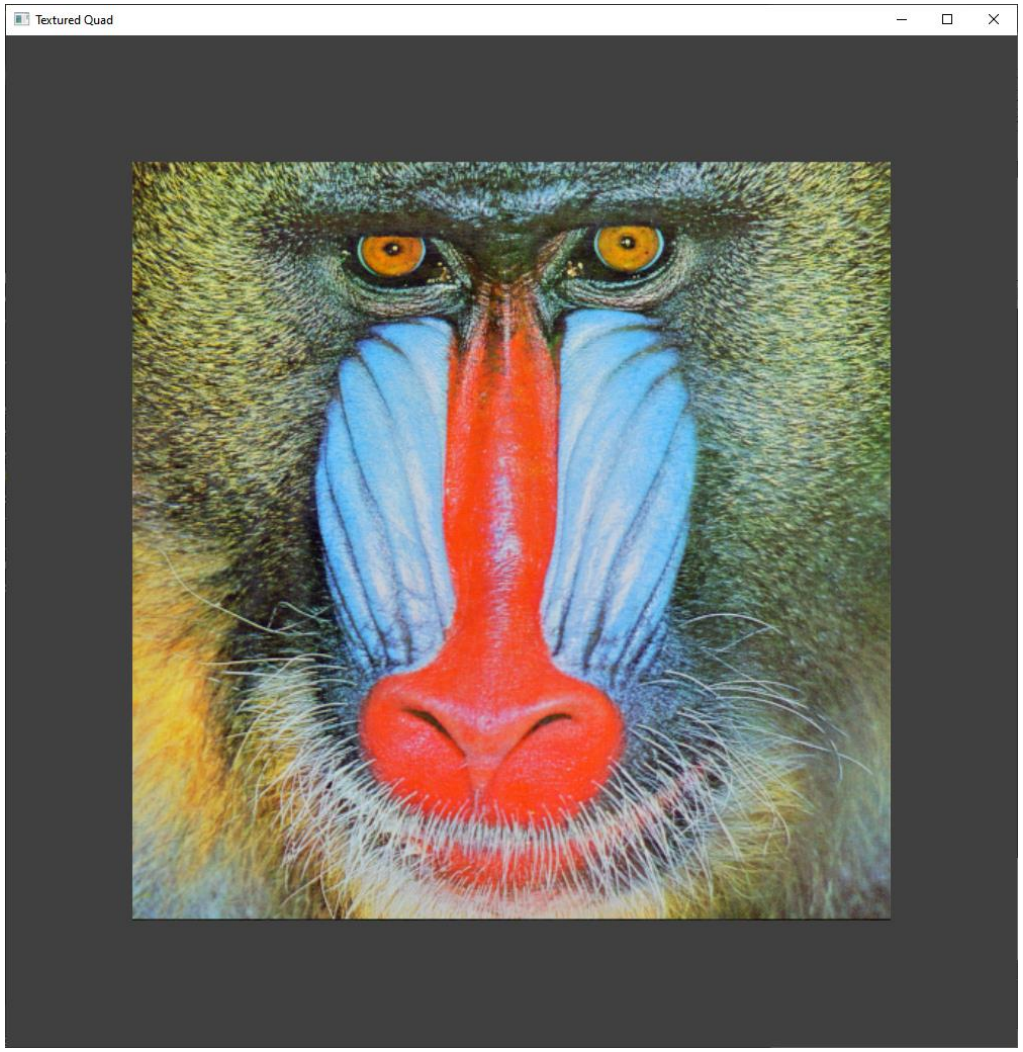
minFilterMode=
GL_NEAREST

maxFilterMode=
GL_NEAREST



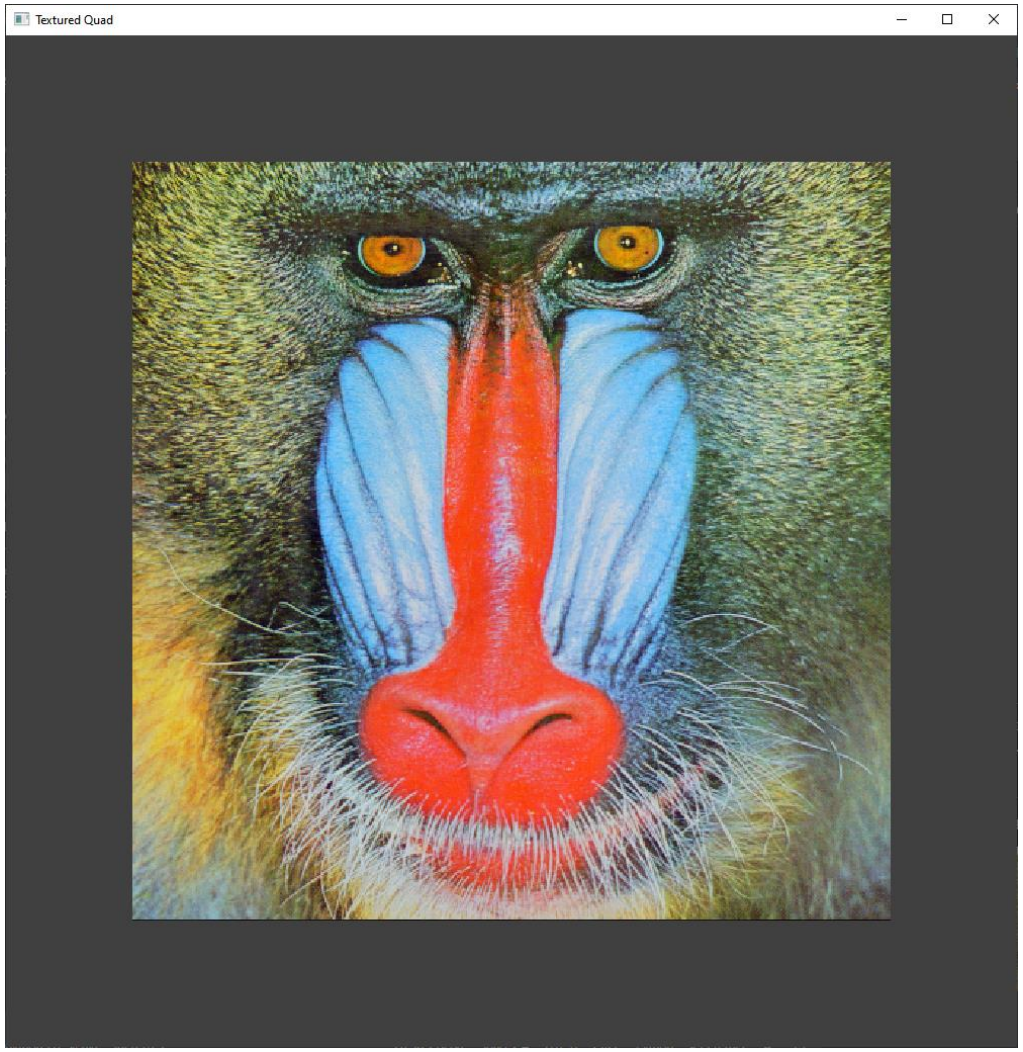
minFilterMode=
GL_LINEAR

maxFilterMode=
GL_LINEAR



minFilterMode=
GL_NEAREST

maxFilterMode=
GL_NEAREST



Controlador

- Keys
- Mouse
- Actions

Controlador

- Keys
- Mouse
- Actions

```
on_key(window, key, scancode, action, mods)
```

- Función usada para recibir las teclas
- Tiene que ser asignada posteriormente con:

```
glfw.set_key_callback(window, on_key)
```

donde `window` es la ventana donde se esta trabajando.

```
glfw.get_key(window, key)
```

- Función que retorna el ultimo estado reportado de la tecla especificada en la ventana especificada

Keys

- A-Z: `glfw.KEY_A - glfw.KEY_Z`
- 0-9: `glfw.KEY_0 - glfw.KEY_9`
- Flechas (◀ ▶ ▲ ▼):

  `glfw.KEY_LEFT, glfw.KEY_RIGHT,`
  `glfw.KEY_UP, glfw.KEY_DOWN`
- Barra Espaciadora: `glfw.KEY_SPACE`
- Tecla Escape (Esc): `glfw.KEY_ESCAPE`

Pueden ver más teclas en: [GLFW Keyboard Keys]

Controlador

- Keys
- Mouse
- Actions

```
cursor_pos_callback(window, x, y)
```

- Función usada para recibir la posición del cursor
- Tiene que ser asignada posteriormente con:

```
glfw.set_cursor_pos_callback(window, cursor_pos_callback)
```

donde `window` es la ventana donde se esta trabajando.

```
mouse_button_callback(window, button, action, mods)
```

- Función usada para recibir los botones del mouse
- Tiene que ser asignada posteriormente con:

  ```
  glfw.set_mouse_button_callback(window, mouse_button_callback)
  ```

  donde `window` es la ventana donde se esta trabajando.
- Click izquierdo: `glfw.MOUSE_BUTTON_1`
- Click derecho: `glfw.MOUSE_BUTTON_2`
- Scroll click (rueda del mouse): `glfw.MOUSE_BUTTON_3`

```
scroll_callback(window, x, y)
```

- Función usada para recibir el desplazamiento de la rueda del mouse
- Tiene que ser asignada posteriormente con:

```
glfw.set_scroll_callback(window, scroll_callback)
```

donde `window` es la ventana donde se esta trabajando.

Controlador

- Keys
- Mouse
- Actions

Actions

- `glfw.PRESS` : La tecla fue presionada
- `glfw.RELEASE`: La tecla fue soltada (tiene que haber sido presionada primero)
- `glfw.REPEAT`: La tecla se mantuvo presionada hasta que se repitió

Ejercicio

- Modelar usando grafo de escena y texturas un árbol del juego Minecraft como se muestra en la imagen
- Para esto modifique el archivo

`ejercicio_texture_minecraft_tree.py`

- (Propuesto) también modele un personaje

que pueda moverse por la escena usando su teclado