Transformaciones en OpenGL

Auxiliar N°3

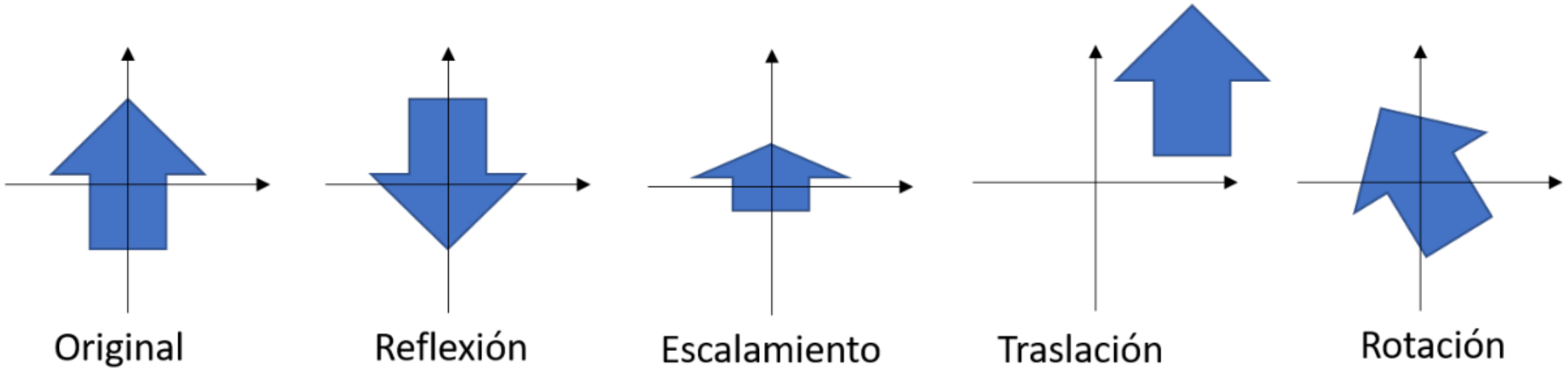
CC3501 – Modelación y Computación Gráfica para Ingenieros

Pablo Pizarro R. @ppizarror.com

Contenidos de hoy

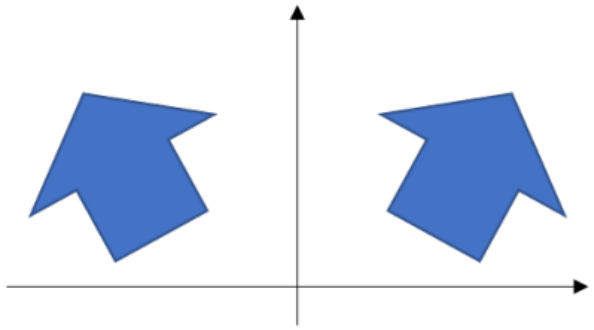
- Transformaciones geométricas.
- Aplicaciones de las matrices de transformación a un problema, usando conceptos de objetos en Python.

Transformaciones geométricas

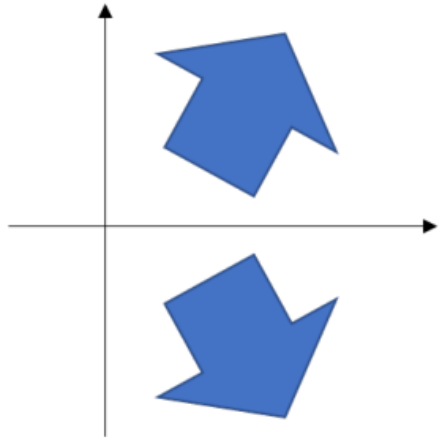


Transformaciones geométricas

- Todas las transformaciones geométricas se pueden representar a través de una matriz de transformación, que modifica el estado de cada vértice del modelo.
- Reflexión:



$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformaciones geométricas

- Escalamiento: α modifica el eje x, β el y, γ el z.

$$P' = S(\alpha, \beta, \gamma)P$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Transformaciones geométricas

- Rotación

Rotación en torno a eje Z

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotación en torno a eje Y

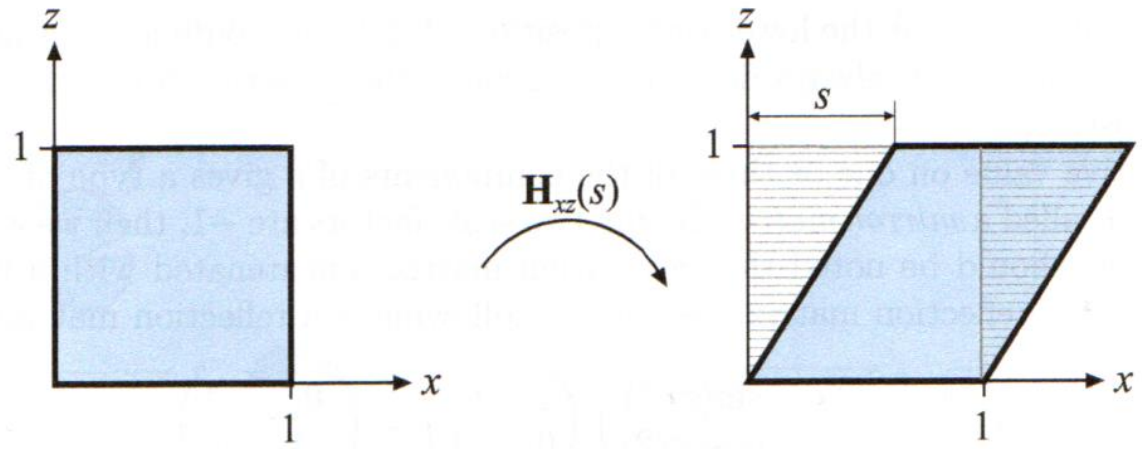
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Rotación en torno a eje Y

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Transformaciones geométricas

- Shearing: Distorsión



$$H_{xz}(s) = \begin{bmatrix} 1 & 0 & s \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformaciones geométricas

- Traslación: También se puede representar en forma matricial usando coordenadas homogéneas.

$$M_T = \left[\begin{array}{c|c} I & T \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Transformaciones geométricas

- Ahora que sabemos cómo se define una transformación (Matriz que opera sobre los vértices). ¿Cómo se usa en OpenGL?

Transformaciones geométricas

- Ahora que sabemos cómo se define una transformación (Matriz que opera sobre los vértices). ¿Cómo se usa en OpenGL?

1) Definir la matriz usando numpy
2) Con OpenGL:
    1) Se carga el modelo con OpenGL, usando el VAO
    2) Se cargan las transformaciones geométricas (matriz)

Transformaciones geométricas

```
def drawShape(shaderProgram, shape, transform):
    # Binding the proper buffers
    glBindVertexArray(shape.vao)
    glBindBuffer(GL_ARRAY_BUFFER, shape.vbo)
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, shape.ebo)

    # updating the new transform attribute
    glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1, GL_FALSE, transform)

    # Describing how the data is stored in the VBO
    position = glGetAttribLocation(shaderProgram, "position")
    glVertexAttribPointer(position, 3, GL_FLOAT, GL_FALSE, 24, ctypes.c_void_p(0))
    glEnableVertexAttribArray(position)

    color = glGetAttribLocation(shaderProgram, "color")
    glVertexAttribPointer(color, 3, GL_FLOAT, GL_FALSE, 24, ctypes.c_void_p(12))
    glEnableVertexAttribArray(color)

    # This line tells the active shader program to render the active element buffer with the given
    size
    glDrawElements(GL_TRIANGLES, shape.size, GL_UNSIGNED_INT, None)
```

Transformaciones geométricas

```
def drawShape(shaderProgram, shape, transform):
    # Binding the proper buffers
    glBindVertexArray(shape.vao)
    glBindBuffer(GL_ARRAY_BUFFER, shape.vbo)
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, shape.ebo)

    # updating the new transform attribute
    glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1, GL_FALSE, transform)

    # Describing how the data is stored in the VBO
    position = glGetAttribLocation(shaderProgram, "position")
    glVertexAttribPointer(position, 3, GL_FLOAT, GL_FALSE, 24, ctypes.c_void_p(0))
    glEnableVertexAttribArray(position)

    color = glGetAttribLocation(shaderProgram, "color")
    glVertexAttribPointer(color, 3, GL_FLOAT, GL_FALSE, 24, ctypes.c_void_p(12))
    glEnableVertexAttribArray(color)

    # This line tells the active shader program to render the active element buffer with the given
    size
    glDrawElements(GL_TRIANGLES, shape.size, GL_UNSIGNED_INT, None)
```

Cargamos el objeto

Aplicamos la matriz de transformación

Transformaciones geométricas

- Si yo quiero aplicar N transformaciones seguidas, ¿tengo que llamar N veces la función para aplicar la transformación?

Transformaciones geométricas

- Si yo quiero aplicar N transformaciones seguidas, ¿tengo que llamar N veces la función para aplicar la transformación?
  - R: No, dado que las transformaciones son matrices uno puede multiplicar todas las matrices a operar sobre un vértice para obtener una única matriz final de transformación.

```
# Create transform matrix
transform = tr.matmul([
    tr.rotationZ(controller.theta),
    tr.translate(controller.x, controller.y, 0.0)
])
```

Transformaciones geométricas

- ¿Cómo hacer las matrices de transformación?

Transformaciones geométricas

- ¿Cómo hacer las matrices de transformación?

1) Crearlas a mano con numpy
2) Usar alguna librería o funciones adicionales: Usar **transform.py**

   Esta librería permite definir matrices de transformación de manera sencilla.

Transformaciones geométricas

- ¿Cómo hacer las matrices de transformación?: Transform.py

```
15  def uniformScale(s):
16      return np.array([
17          [s, 0, 0, 0],
18          [0, s, 0, 0],
19          [0, 0, s, 0],
20          [0, 0, 0, 1]], dtype=np.float32).T
21
22
23  def scale(sx, sy, sz):
24      return np.array([
25          [sx, 0, 0, 0],
26          [0, sy, 0, 0],
27          [0, 0, sz, 0],
28          [0, 0, 0, 1]], dtype=np.float32).T
29
30
31  def rotationX(theta):
32      sin_theta = np.sin(theta)
33      cos_theta = np.cos(theta)
34
35      return np.array([
```

Transformaciones geométricas

- Problema de hoy:

1) Comenzando con el archivo ex_quad.py y entendiendo como se aplican las transformaciones (archivos ex_transformations.py y ex_4shapes.py), asígnele movimiento discreto a un cuadrado.
    1) Al presionar las flechas del teclado, el cuadrado se debe mover 0.1 en la dirección indicada.
    2) Al presionar 'espacio', el cuadrado debe comenzar a rotar sobre sí mismo (y NO sobre el 0,0 del centro de la ventana).

Transformaciones geométricas

Problema de hoy (1):

- El objetivo es entender cómo funcionan los eventos
- Los eventos se leen en cada ciclo del programa
- Al apretar un botón el programa debe ejecutar alguna acción, en este caso, mover el cuadrado corresponde a sumar a alguna variable de posición del cuadrado cierta constante.

Transformaciones geométricas

Problema de hoy (1):

```
32  # noinspection PyUnusedLocal
33  def on_key(window, key, scancode, action, mods):
34      global controller
35
36      # Keep pressed buttons
37      if action == glfw.REPEAT or action == glfw.PRESS:
38          if key == glfw.KEY_LEFT:
39              controller.x -= 0.1
40          elif key == glfw.KEY_RIGHT:
41              controller.x += 0.1
42          elif key == glfw.KEY_UP:
43              controller.y += 0.1
44          elif key == glfw.KEY_DOWN:
45              controller.y -= 0.1
46
47      if action != glfw.PRESS:
48          return
49
50      if key == glfw.KEY_SPACE:
51          controller.rotate = not controller.rotate
52
53      elif key == glfw.KEY_1:
54          controller.fillPolygon = not controller.fillPolygon
55
56      elif key == glfw.KEY_ESCAPE:
```

Transformaciones geométricas

Problema de hoy (1):

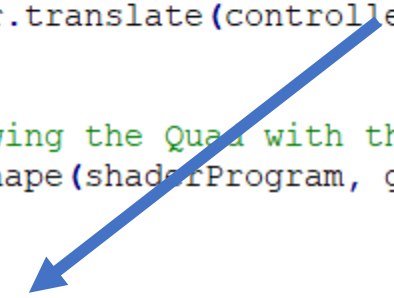
```
32  # noinspection PyUnusedLocal
33  def on_key(window, key, scancode, action, mods):
34      global controller
35
36      # Keep pressed buttons
37      if action == glfw.REPEAT or action == glfw.PRESS:
38          if key == glfw.KEY_LEFT:
39              controller.x -= 0.1
40          elif key == glfw.KEY_RIGHT:
41              controller.x += 0.1
42          elif key == glfw.KEY_UP:
43              controller.y += 0.1
44          elif key == glfw.KEY_DOWN:
45              controller.y -= 0.1
46
47      if action != glfw.PRESS:
48          return
49
50      if key == glfw.KEY_SPACE:
51          controller.rotate = not controller.rotate
52
53      elif key == glfw.KEY_1:
54          controller.fillPolygon = not controller.fillPolygon
55
56      elif key == glfw.KEY_ESCAPE:
```

On_key es una función que se ejecuta en cada iteración del programa. Al apretar KEY_LEFT se suma a una variable global (controller) la posición en x, incrementando 0.1 cada vez que este evento se lee.

Transformaciones geométricas

Problema de hoy (1):

```
218         # Clearing the screen in both, color and depth
219         glClear(GL_COLOR_BUFFER_BIT)
220
221         # theta is modified an amount proportional to the time spent in a loop iteration
222         if controller.rotate:
223             controller.theta += dt
224
225         # Create transform matrix
226         transform = tr.matmul([
227             tr.rotationZ(controller.theta),
228             tr.translate(controller.x, controller.y, 0.0)
229         ])
230
231         # Drawing the Quad with the given transformation
232         drawShape(shaderProgram, gpuQuadBlack, transform)
```



En cada ciclo, se crea una matriz de transformación usando estas variables globales de posición

```
19  # A class to store the application control
20  class Controller:
21      x = 0.0
22      y = 0.0
23      theta = 0.0
24      rotate = False
25      fillPolygon = True
26
```

Transformaciones geométricas

- Problema de hoy:

2) Implemente funciones para generar las gpuShapes de dos cuadrados, uno blanco y uno negro

Transformaciones geométricas

- Problema de hoy:

3) Utilizando las funciones anteriores, dibuje un tablero de ajedrez

Muchas gracias por su atención

¿Preguntas?