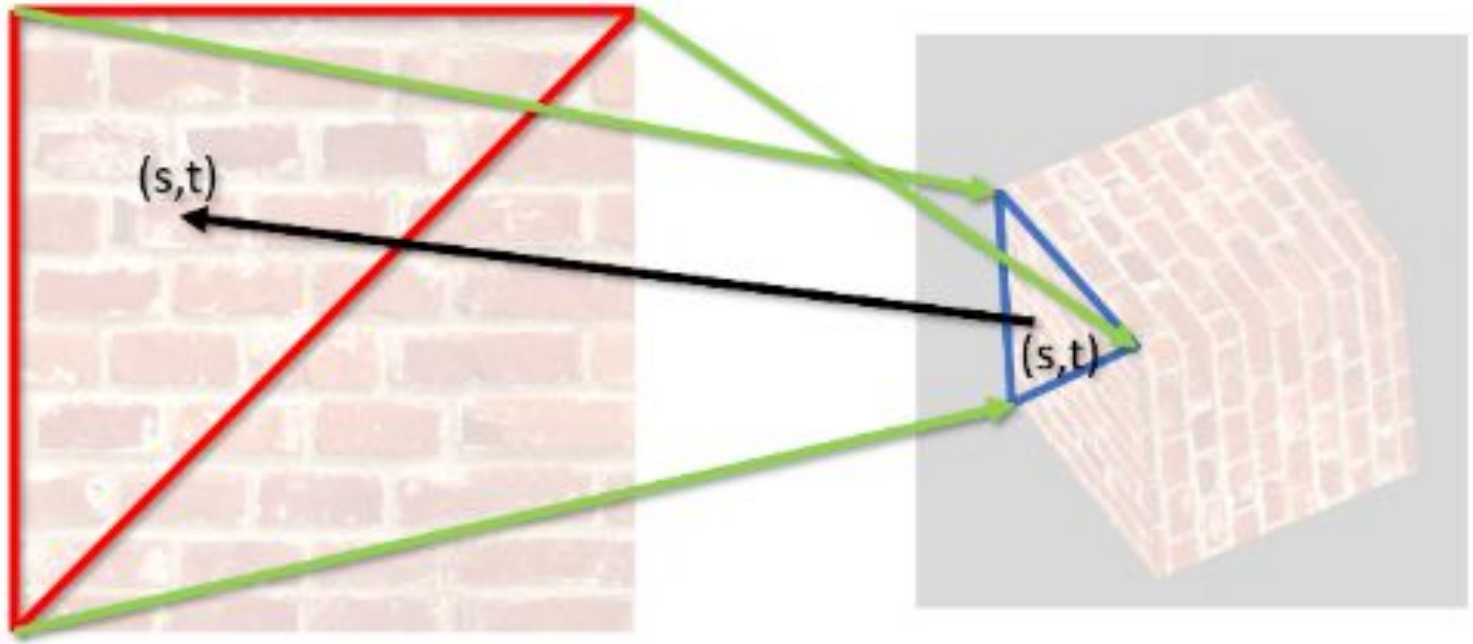
Auxiliar 5 - Texturas

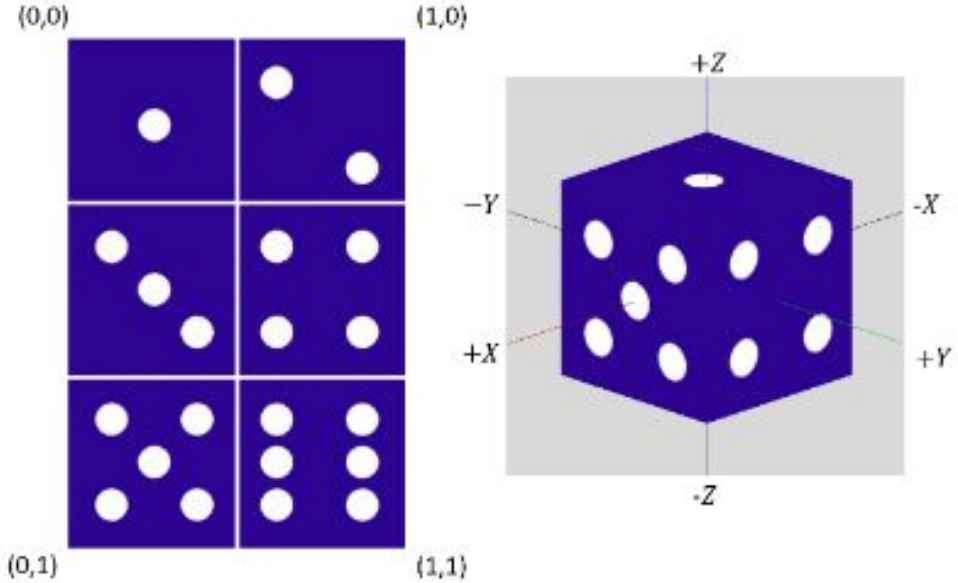
Nelson Marambio

CC3501 - Modelación y computación gráfica para ingenieros

Texturas



Texturas en OpenGl



Número 4 queda definido por:

```
vertices = [
...
# positions        # tex coords
-0.5,  0.5, -0.5, 1/2, 2/3,
 0.5,  0.5, -0.5,   1, 2/3,
 0.5,  0.5,  0.5,   1, 1/3,
-0.5,  0.5,  0.5, 1/2, 1/3,
...
]
```

Pregunta 1

Junto con python y OpenGL, cree un sprite animado de un caballero, utilizando la imagen adjunta en la carpeta ’Sprites’, que corresponde a un texture atlas. Para ello tome el archivo ’ex_texture_boo.py’ y modifíquelo para lograr lo pedido. Al presionar la flecha derecha e izquierda del teclado, el caballero debe moverse hacia la derecha e izquierda respectivamente, y además cambiar la textura que se está utilizando, para así formar la animación. Para extraer las diferentes partes del sprite, modifique el vertex shader, haciendo que reciba un número que indique el frame de la textura que quiera utilizar. Para ello inspirese en easy_shaders y cree su propio código.

Pregunta 1 - Solución

Lo que haremos será tener una variable que guarde el índice de la imagen que queremos poner en nuestra figura. Esta variable será modificada por el input del teclado, permitiendo que cada vez que se mueva nuestra figura, esta cambie la textura asociada. Luego, este índice será enviado al vertex shader, quien asociará el trozo de la imagen a la figura, así simulando el movimiento del caballero. Como el vertex shader se encargará de asociar la textura a la figura, entonces no debemos crear índices de textura al crear la figura.

Pregunta 2 - Solución

Primero creamos una función que nos entregue solo las posiciones de los vértices de un cuadrado (sin color ni textura):

```
def createQuad():
    # Defining locations and texture coordinates for each vertex of the shape
    vertices = [
        #   positions
        -0.5, -0.5,
        0.5, -0.5,
        0.5,  0.5,
        -0.5,  0.5, ]
    # Defining connections among vertices
    # We have a triangle every 3 indices specified
    indices = [
        0, 1, 2,
        2, 3, 0]
    return bs.Shape(vertices, indices)
```

Pregunta 2 - Solución

Reemplazamos las gpu shapes creadas, por la nuestra con nuestra textura:

```
# Creating shapes on GPU memory
    shapeKnight = createQuad()
    gpuKnight = GPUShape().initBuffers()
    pipeline.setupVAO(gpuKnight)
    # Definimos donde se encuentra la textura
    thisFilePath = os.path.abspath(__file__)
    thisFolderPath = os.path.dirname(thisFilePath)
    spritesDirectory = os.path.join(thisFolderPath, "Sprites")
    spritePath = os.path.join(spritesDirectory, "sprites.png")
    gpuKnight.texture = es.textureSimpleSetup(
        spritePath, GL_CLAMP_TO_EDGE, GL_CLAMP_TO_EDGE, GL_NEAREST, GL_NEAREST)
    gpuKnight.fillBuffers(shapeKnight.vertices,
                          shapeKnight.indices, GL_STATIC_DRAW)
```

Pregunta 2 - Solución

Modificamos solamente el vertex shader:

```
class ShaderPregunta2(SimpleTextureTransformShaderProgram):
    def __init__(self):
        vertex_shader = """
            #version 130
            uniform mat4 transform;
            uniform float texture_index;
            in vec2 position;
            out vec2 outTexCoords;
            void main(){
                gl_Position = transform * vec4(position, 0, 1.0f);
                if(position.x>0 && position.y>0){
                    outTexCoords = vec2((texture_index + 1)*1/10, 0);}
                else if(position.x<0 && position.y>0){
                    outTexCoords = vec2(texture_index*1/10, 0);}
                else if(position.x>0 && position.y<0){
                    outTexCoords = vec2((texture_index + 1)*1/10, 1);}
                else{
                    outTexCoords = vec2(texture_index*1/10, 1);}}
            """
```

Pregunta 2 - Solución

Modificamos la función setupVAO, debido a que solo mandamos las posiciones de la figura:

```
def setupVAO(self, gpuShape):
        glBindVertexArray(gpuShape.vao)

        glBindBuffer(GL_ARRAY_BUFFER, gpuShape.vbo)
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, gpuShape.ebo)

        # 2d vertices => 2*4 = 8 bytes
        position = glGetAttribLocation(self.shaderProgram, "position")
        glVertexAttribPointer(position, 2, GL_FLOAT,
                              GL_FALSE, 8, ctypes.c_void_p(0))
        glEnableVertexAttribArray(position)

        # Unbinding current vao
        glBindVertexArray(0)
```

Pregunta 2 - Solución

Modificamos el controller para que así se mueva el personaje:

```
# A class to store the application control
class Controller:
    def __init__(self):
        self.fillPolygon = True

#########################################################################
############################
        self.actual_sprite = 1
        self.x = 0.0
#########################################################################
############################
```

Pregunta 2 - Solución

Modificamos la función on_key para que así, al apretar las teclas, modifique las variables añadidas:

```
##########################################################################
    elif key == glfw.KEY_RIGHT:
        controller.x += 0.05
        controller.actual_sprite = (controller.actual_sprite + 1) % 10

    elif key == glfw.KEY_LEFT:
        controller.x -= 0.05
        controller.actual_sprite = (controller.actual_sprite - 1) % 10
##########################################################################
```

Pregunta 2 - Solución

Creamos una instancia de nuestro nuevo shader:

```
#############################################################################
    # A simple shader program with position and texture coordinates as
inputs.
    pipeline = es.ShaderPregunta2()
#############################################################################
```

Pregunta 2 - Solución

Modificamos para que se dibuje nuestra figura, entregando las variables necesarias al shader:

```
###############################################################################################
        glUniformMatrix4fv(glGetUniformLocation(pipeline.shaderProgram, "transform"), 1,
GL_TRUE, tr.matmul([
            tr.translate(controller.x, 0, 0),
            tr.uniformScale(0.5)
        ]))

        glUniform1f(glGetUniformLocation(pipeline.shaderProgram,
                    "texture_index"), controller.actual_sprite)

        pipeline.drawCall(gpuKnight)

###############################################################################################
```

Pregunta 2 - Colisiones

Al caballero de la pregunta anterior, agréguele colisiones a ambos lados de la pantalla. Esto quiere decir que si el caballero llega al final de la pantalla, este no podrá seguir avanzando en esa dirección.

Pregunta 2 - Solución

Lo que haremos será revisar la posición del caballero en cada frame, y si la posición corresponde a alguno de los bordes de la pantalla, entonces no moveremos al caballero.

Para esto agregaremos esta condición en la función on_key.

Pregunta 2 - Solución

```
##########################################################################################################

    elif key == glfw.KEY_RIGHT:
        if controller.x <= 1 - 0.25:
            controller.x += 0.05
            controller.actual_sprite = (controller.actual_sprite + 1) % 10

    elif key == glfw.KEY_LEFT:
        if controller.x >= -1 + 0.25:
            controller.x -= 0.05
            controller.actual_sprite = (controller.actual_sprite - 1) % 10

    ##########################################################################################################
```

Pregunta 2 - Solución

Las condiciones usadas anteriormente, también puede ser utilizada para otros fines, como por ejemplo, para que un jugador pierda si toca el piso, o si choca con algún objeto.