3D graphics – practical 1

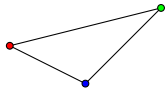
François-Xavier Carton

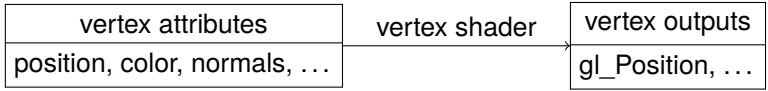
February 1st, 2022

Rendering pipeline (GPU)

1. vertex processing

$\forall vertex$:

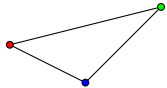




Rendering pipeline (GPU)

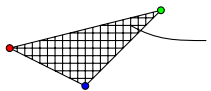
1. vertex processing

$\forall vertex$:



vertex attributes	vertex shader	vertex outputs
position, color, normals, ...	→	gl_Position, ...

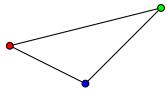
2. rasterization



Rendering pipeline (GPU)

1. vertex processing

$\forall$*vertex* :

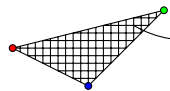


vertex attributes
position, color, normals, . . .

vertex shader

vertex outputs
gl_Position, . . .

2. rasterization

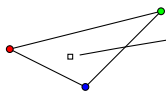


fragment

interpolation

3. fragment processing

$\forall$*fragment* :



fragment inputs

fragment shader

outColor

viewer.py

```
class Shader:
    # compile and link shaders

class Triangle:
    def __init__(self, shader):
        # 1) setup vertex attributes
        # 2) transfer them to GPU

    def draw(self):
        # tell GPU to render the triangle

    def __del__(self):
        # free GPU resources

class Viewer:
    # window creation
    # opengl context creation

    def run(self):
        # rendering loop

    def add(self, *drawables):
        # add object to scene

def main():
    # setup viewer
    # add objects to scene
    # call main loop
```

viewer.py

```
class Shader:
    # compile and link shaders

class Triangle:
    def __init__(self, shader):
        # 1) setup vertex attributes
        # 2) transfer them to GPU

    def draw(self):
        # tell GPU to render the triangle

    def __del__(self):
        # free GPU resources

class Viewer:
    # window creation
    # opengl context creation

    def run(self):
        # rendering loop

    def add(self, *drawables):
        # add object to scene

def main():
    # setup viewer
    # add objects to scene
    # call main loop
```

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

0	0.5	0	0.5	-0.5	0	-0.5	-0.5	0

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

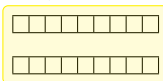
fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Buffer Objects (VBOs):

buffer0

buffer1



CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

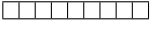
fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Buffer Objects (VBOs):

buffer0

buffer1



CPU

GPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

position | x_0 | y_0 | z_0 | x_1 | y_1 | z_1 | x_2 | y_2 | z_2 |

color | r_0 | g_0 | b_0 | r_1 | g_1 | b_1 | r_2 | g_2 | b_2 |

Vertex Buffer Objects (VBOs):

buffer0 | x_0 | y_0 | z_0 | x_1 | y_1 | z_1 | x_2 | y_2 | z_2 |

buffer1 | | | | | | | | | |

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)

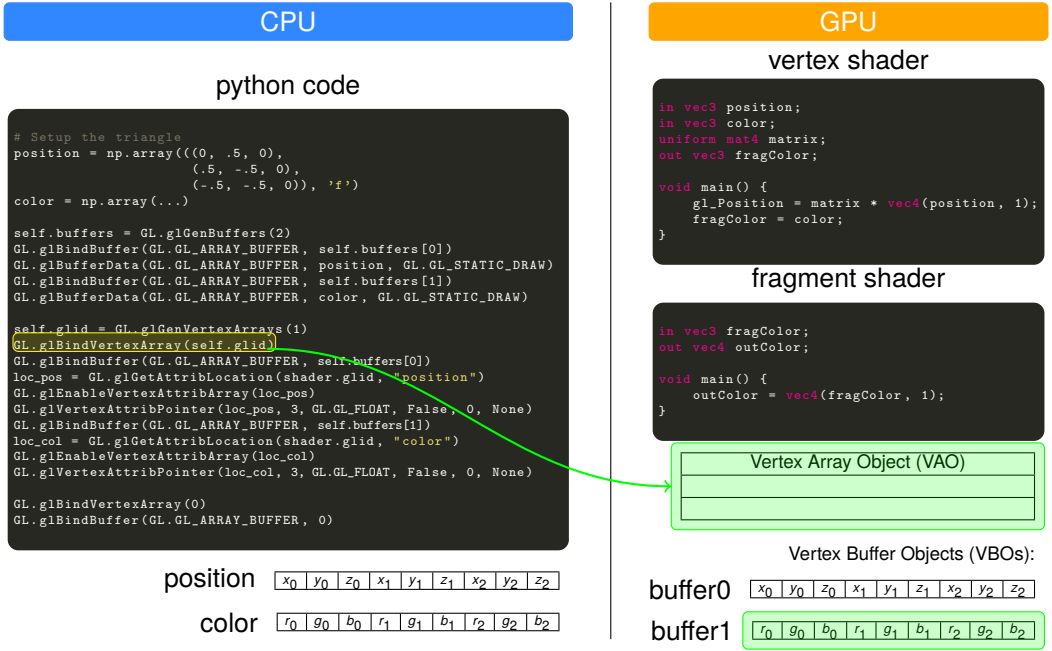
Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2


CPU
python code
# Setup the triangle
position = np.array(((0, .5, 0),
(.5, -.5, 0),
(-.5, -.5, 0)), 'f')
color = np.array(...)
self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)
self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
position
x0 y0 z0 x1 y1 z1 x2 y2 z2
color
r0 g0 b0 r1 g1 b1 r2 g2 b2
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
Vertex Array Object (VAO)
Vertex Buffer Objects (VBOs):
buffer0
x0 y0 z0 x1 y1 z1 x2 y2 z2
buffer1
r0 g0 b0 r1 g1 b1 r2 g2 b2

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)

Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)

to loc_pos, from buffer0, 3 floats/vertex

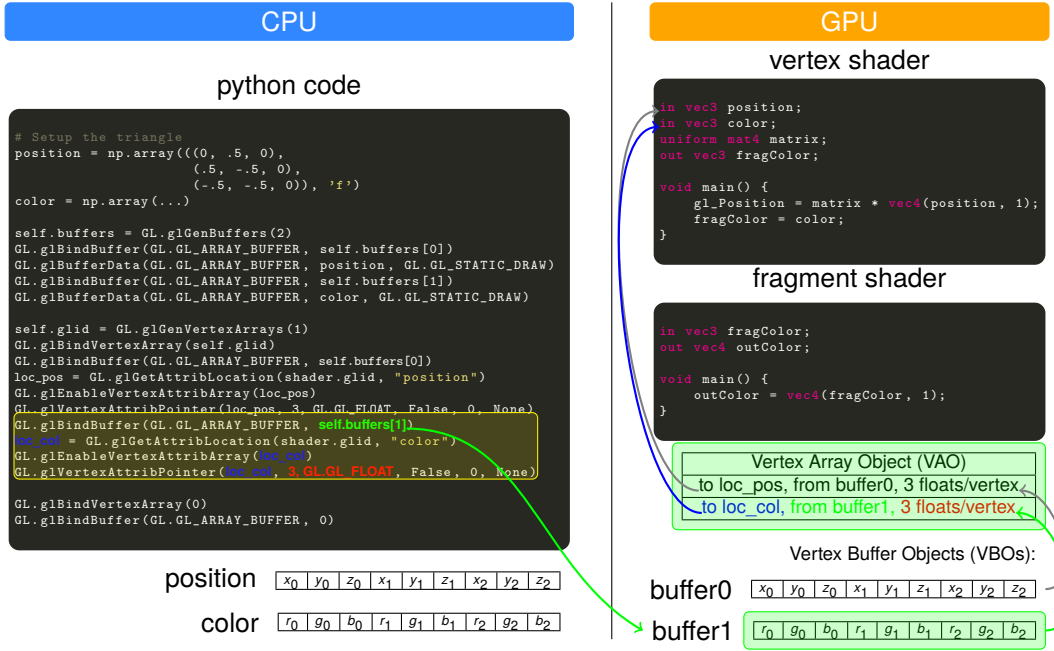
Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2


CPU
python code
# Setup the triangle
position = np.array(((0, .5, 0),
(.5, -.5, 0),
(-.5, -.5, 0)), 'f')
color = np.array(...)
self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)
self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
position
color
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
Vertex Array Object (VAO)
to loc_pos, from buffer0, 3 floats/vertex
to loc_col, from buffer1, 3 floats/vertex
Vertex Buffer Objects (VBOs):
buffer0
buffer1

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)

- to loc_pos, from buffer0, 3 floats/vertex
- to loc_col, from buffer1, 3 floats/vertex

Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

CPU

python code

```
# Setup the triangle
position = np.array(((0, .5, 0),
                     (.5, -.5, 0),
                     (-.5, -.5, 0)), 'f')
color = np.array(...)

self.buffers = GL.glGenBuffers(2)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, position, GL.GL_STATIC_DRAW)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
GL.glBufferData(GL.GL_ARRAY_BUFFER, color, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, 0, None)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[1])
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, 0, None)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

position

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

color

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)
to loc_pos, from buffer0, 3 floats/vertex
to loc_col, from buffer1, 3 floats/vertex

Vertex Buffer Objects (VBOs):

buffer0

x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

buffer1

r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2

viewer.py

```
class Shader:
    # compile and link shaders

class Triangle:
    def __init__(self, shader):
        # 1) setup vertex attributes
        # 2) transfer them to GPU

    def draw(self):
        # tell GPU to render the triangle

    def __del__(self):
        # free GPU resources

class Viewer:
    # window creation
    # opengl context creation

    def run(self):
        # rendering loop

    def add(self, *drawables):
        # add object to scene

def main():
    # setup viewer
    # add objects to scene
    # call main loop
```

CPU

python code

```
# Draw the triangle
GL.glUseProgram(self.shader.glid)

matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)

GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
```

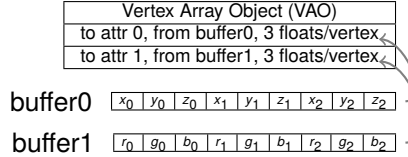
GPU

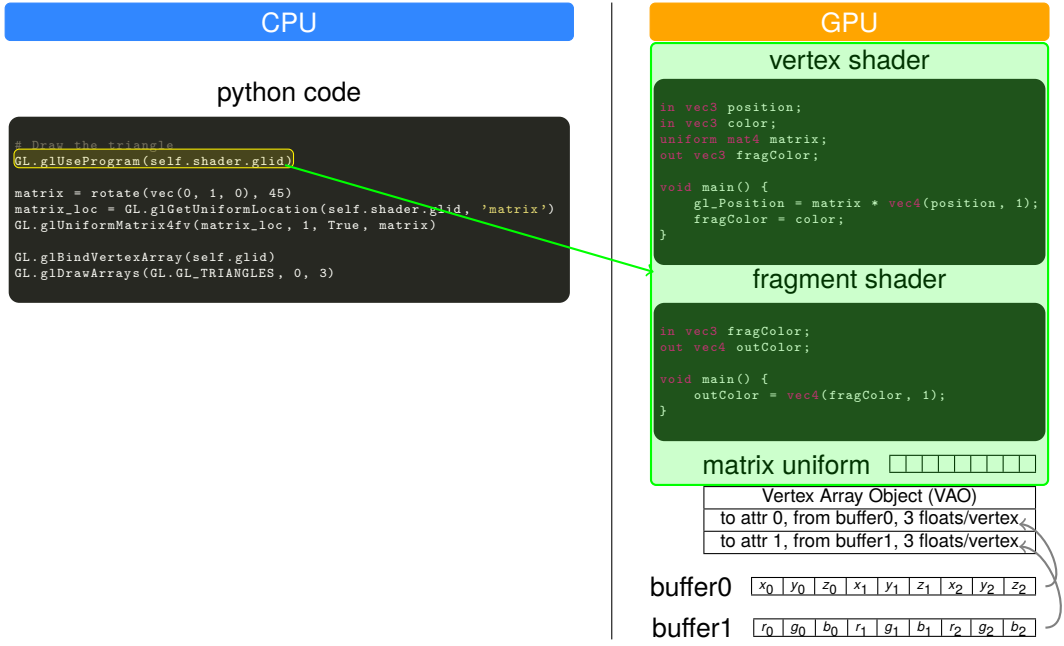
vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```




CPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2

CPU

python code

```
# Draw the triangle
GL.glUseProgram(self.shader.glid)

matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)

GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
```

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

matrix uniform

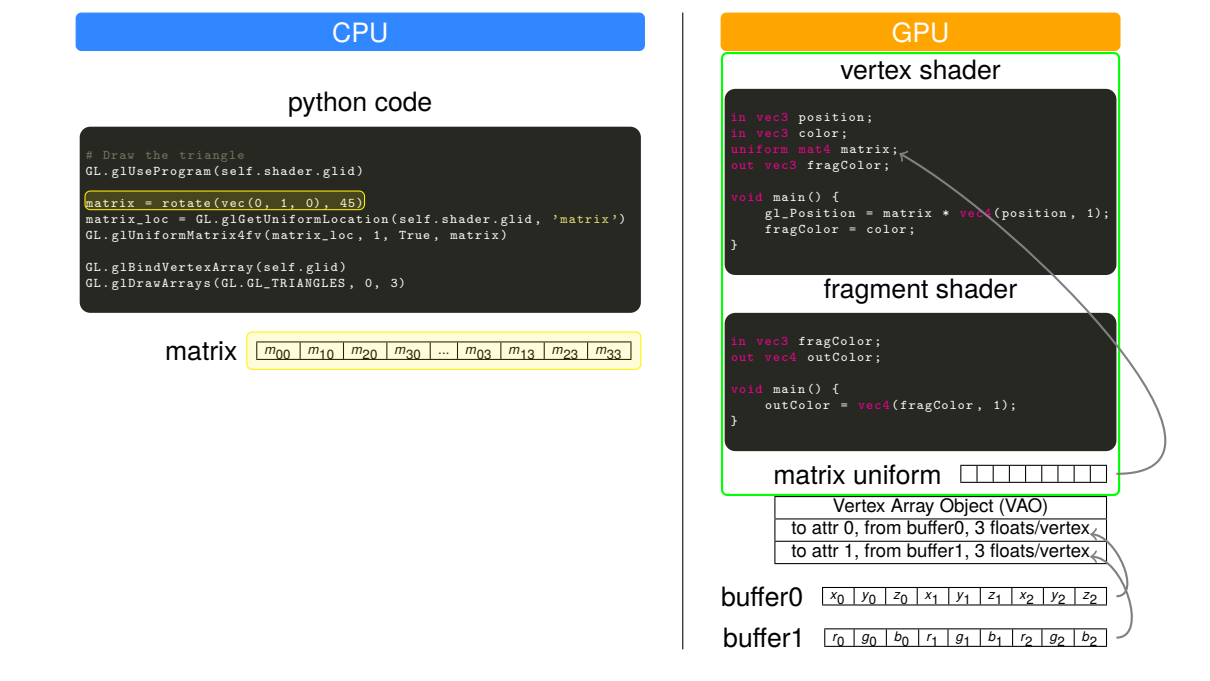
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex

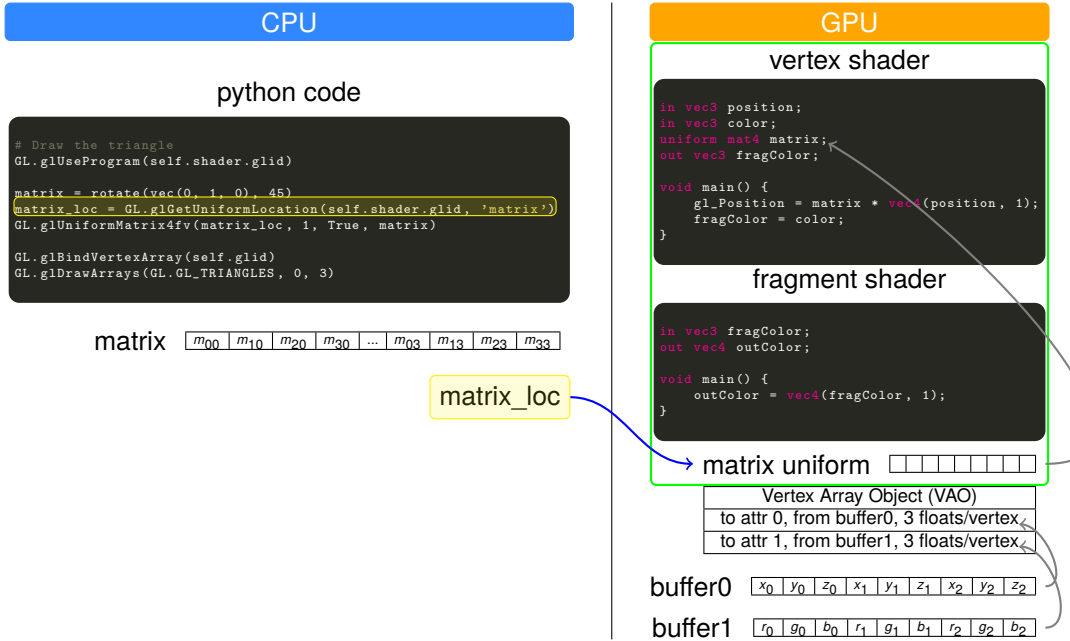
buffer0

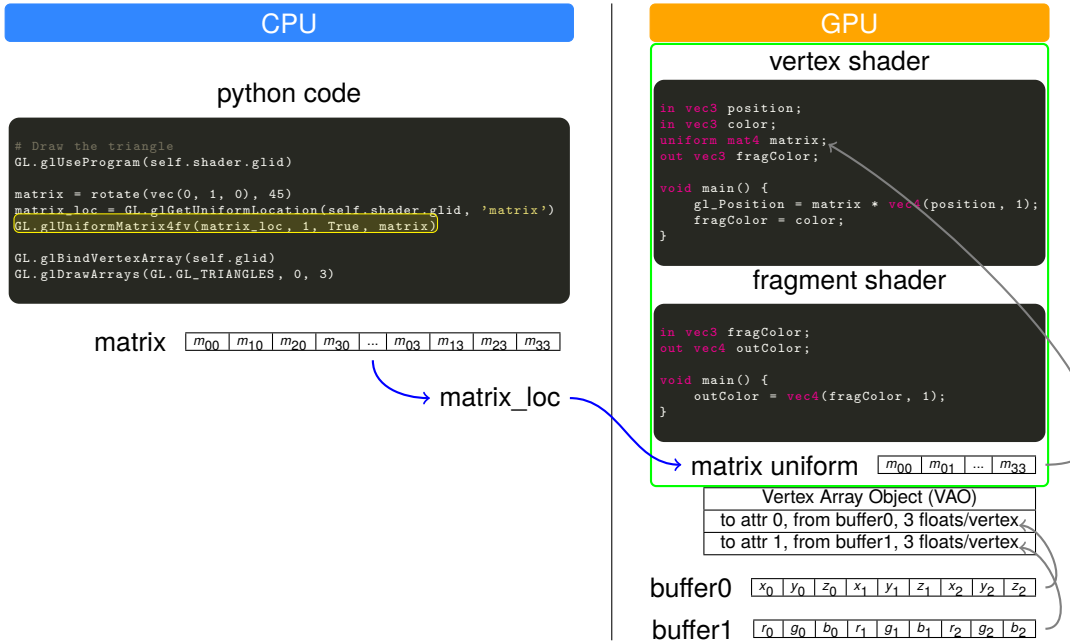
x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2

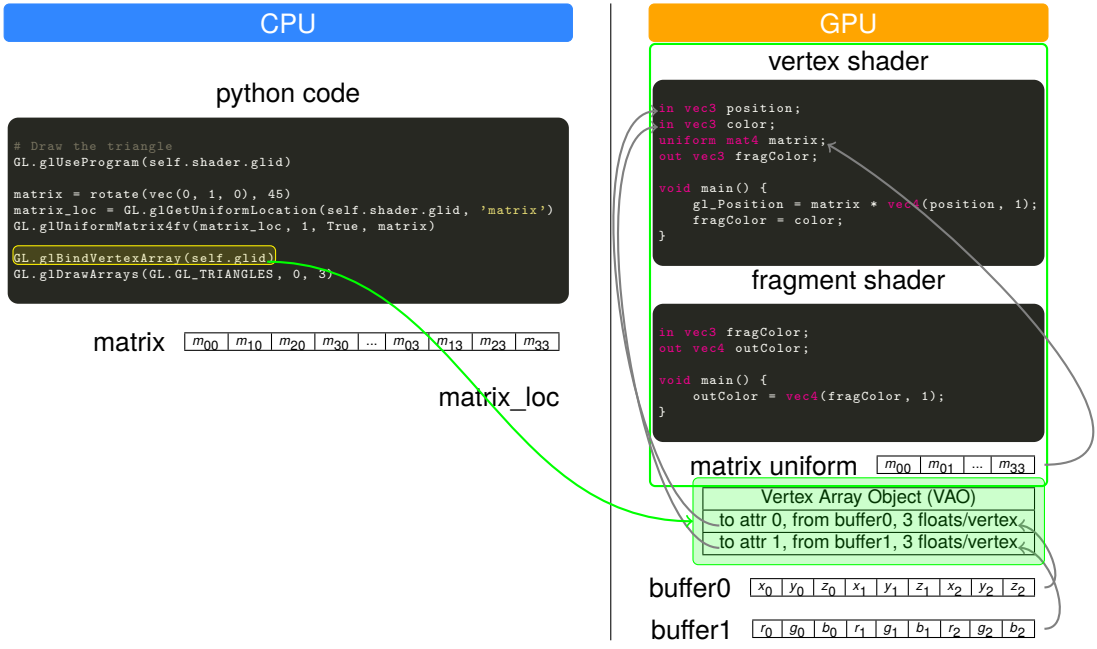
buffer1

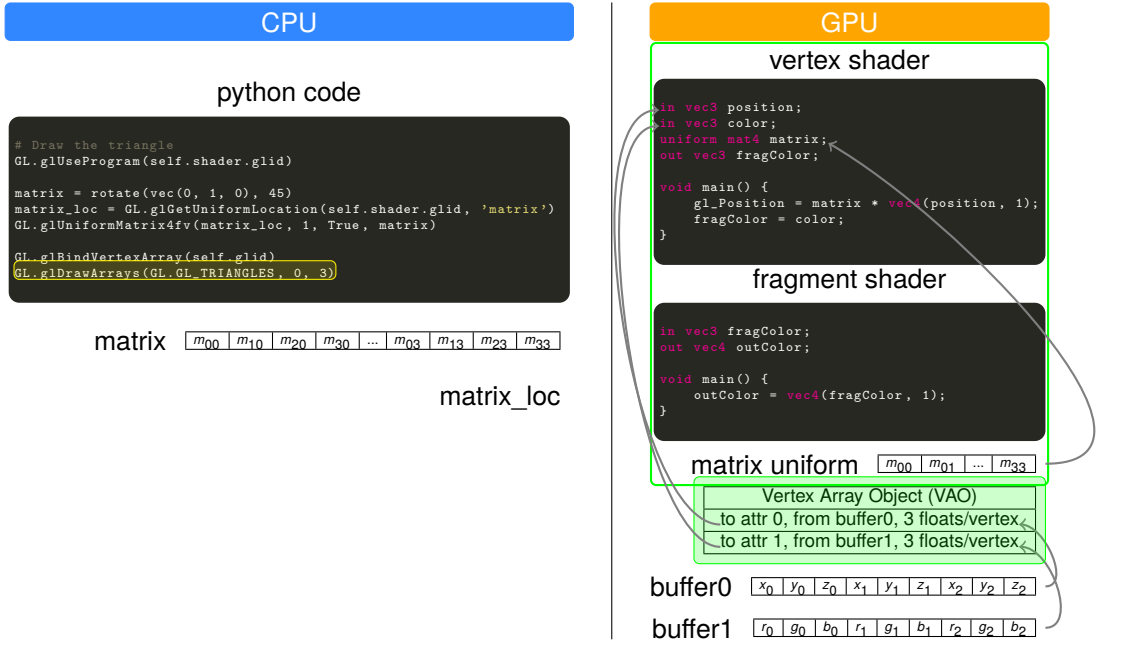
r_0	g_0	b_0	r_1	g_1	b_1	r_2	g_2	b_2


CPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
matrix
m_{00} m_{10} m_{20} m_{30} ... m_{03} m_{13} m_{23} m_{33}
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2


CPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
matrix
m_{00} m_{10} m_{20} m_{30} ... m_{03} m_{13} m_{23} m_{33}
matrix_loc
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2

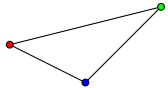

CPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
matrix
m_{00} m_{10} m_{20} m_{30} ... m_{03} m_{13} m_{23} m_{33}
matrix_loc
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
m_{00} m_{01} ... m_{33}
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2


CPU
GPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
matrix
m_{00} m_{10} m_{20} m_{30} ... m_{03} m_{13} m_{23} m_{33}
matrix_loc
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
m_{00} m_{01} ... m_{33}
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2


CPU
python code
# Draw the triangle
GL.glUseProgram(self.shader.glid)
matrix = rotate(vec(0, 1, 0), 45)
matrix_loc = GL.glGetUniformLocation(self.shader.glid, 'matrix')
GL.glUniformMatrix4fv(matrix_loc, 1, True, matrix)
GL.glBindVertexArray(self.glid)
GL.glDrawArrays(GL.GL_TRIANGLES, 0, 3)
matrix
m_{00} m_{10} m_{20} m_{30} ... m_{03} m_{13} m_{23} m_{33}
matrix_loc
GPU
vertex shader
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
void main() {
gl_Position = matrix * vec4(position, 1);
fragColor = color;
}
fragment shader
in vec3 fragColor;
out vec4 outColor;
void main() {
outColor = vec4(fragColor, 1);
}
matrix uniform
m_{00} m_{01} ... m_{33}
Vertex Array Object (VAO)
to attr 0, from buffer0, 3 floats/vertex
to attr 1, from buffer1, 3 floats/vertex
buffer0
x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2
buffer1
r_0 g_0 b_0 r_1 g_1 b_1 r_2 g_2 b_2

Rendering pipeline (GPU)

1. vertex processing



vertex attributes

position	(-2, 0, 0)
color	(1, 0, 0)
position	(2, 1, 0)
color	(0, 1, 0)
position	(0, -1, 0)
color	(0, 0, 1)

vertex shader

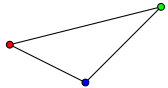
```
in vec3 position;
in vec3 color;
out vec3 fragColor;

void main() {
    gl_Position = position;
    fragColor = color;
}
```

vertex out variables

gl_Position	
fragColor	
gl_Position	
fragColor	
gl_Position	
fragColor	

Rendering pipeline (GPU)

1. vertex processing



vertex attributes

position	(-2, 0, 0)
color	(1, 0, 0)
position	(2, 1, 0)
color	(0, 1, 0)
position	(0, -1, 0)
color	(0, 0, 1)

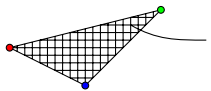
vertex shader

```
in vec3 position;
in vec3 color;
out vec3 fragColor;

void main() {
    gl_Position = position;
    fragColor = color;
}
```

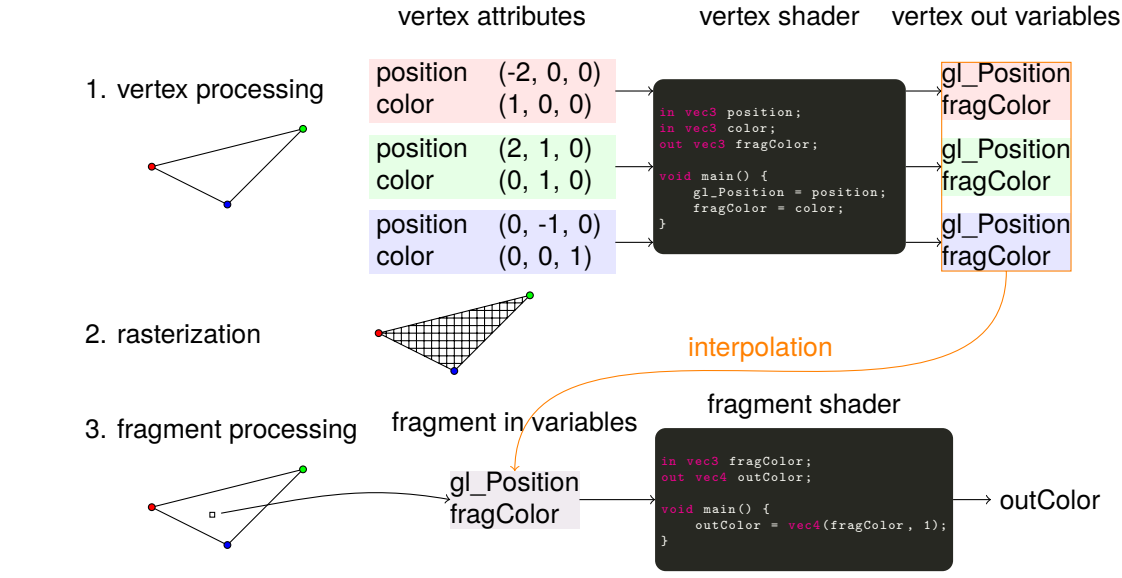
vertex out variables

gl_Position	
fragColor	
gl_Position	
fragColor	
gl_Position	
fragColor	

2. rasterization

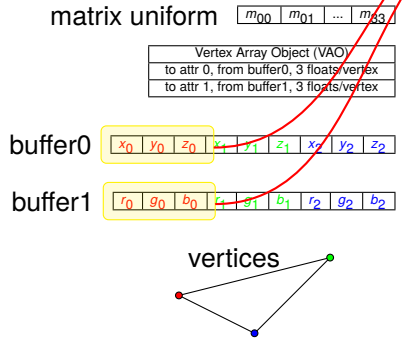


Rendering pipeline (GPU)



vertex processing

one execution of the *vertex shader* per vertex, in parallel



vertex shader - instance 0

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 1

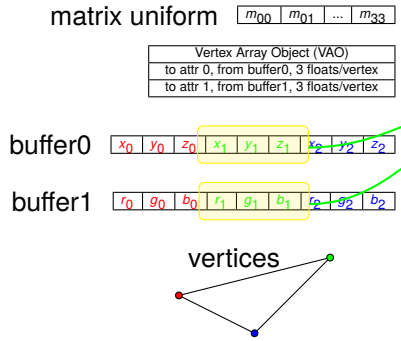
```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 2

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex processing

one execution of the *vertex shader* per vertex, in parallel



vertex shader - instance 0

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 1

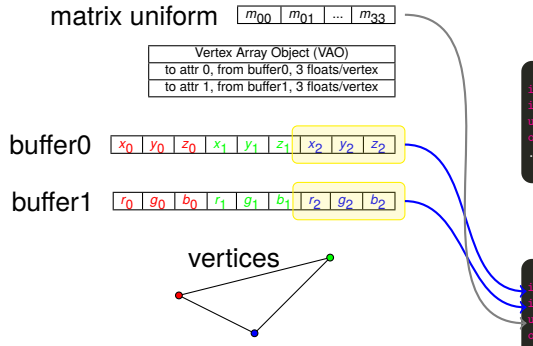
```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 2

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex processing

one execution of the *vertex shader* per vertex, in parallel



vertex shader - instance 0

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 1

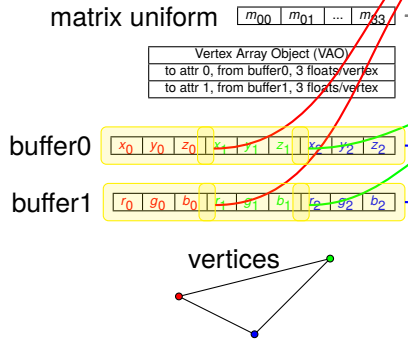
```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 2

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex processing

one execution of the *vertex shader* per vertex, **in parallel**



vertex shader - instance 0

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 1

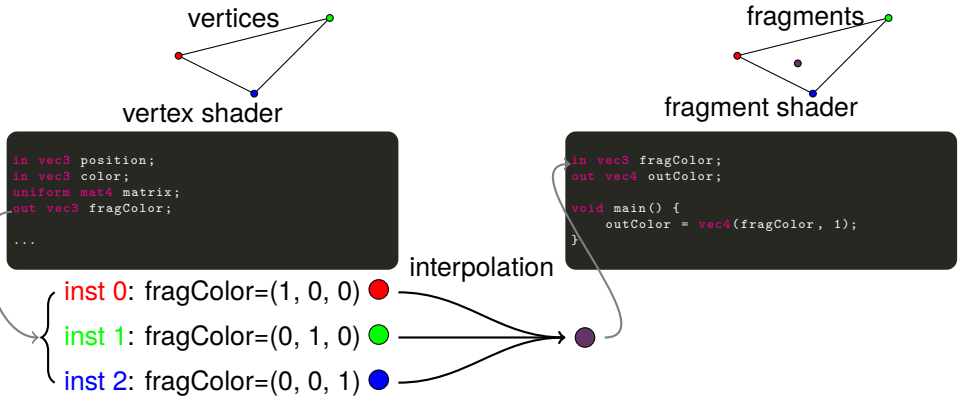
```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

vertex shader - instance 2

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;
...
```

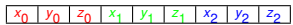
fragment processing

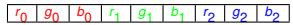
one execution of the *fragment shader*
per fragment in the triangle, in parallel



Bonus: alternative layouts of the attributes

- one buffer per attribute

position 

color 

- **interlaced attributes**

poscol 

Bonus: alternative layouts of the attributes

- one buffer per attribute

position | x_0 | y_0 | z_0 | x_1 | y_1 | z_1 | x_2 | y_2 | z_2 |

color | r_0 | g_0 | b_0 | r_1 | g_1 | b_1 | r_2 | g_2 | b_2 |

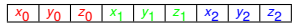
- **interlaced attributes**

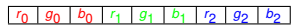
poscol | x_0 | y_0 | z_0 | r_0 | g_0 | b_0 | x_1 | y_1 | z_1 | r_1 | g_1 | b_1 | x_2 | y_2 | z_2 | r_2 | g_2 | b_2 |

stride nb of bytes per vertex (here $6 \times sizeof(FLOAT) == 24$)

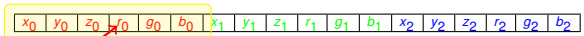
Bonus: alternative layouts of the attributes

- one buffer per attribute

position 

color 

- **interlaced attributes**

poscol 

stride nb of bytes per vertex (here $6 \times sizeof(FLOAT) == 24$)

offset (color) nb of bytes before color attributes (here $3 \times sizeof(FLOAT) == 12$)
this integer (12) is actually cast to a *pointer* to the "address" 12
(legacy to old versions)

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,   0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                          # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

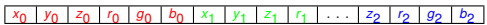
```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol



GPU

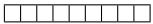
vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

buffer0



CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,   0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                  # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

buffer0

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

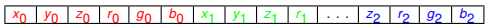
buffer0

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol



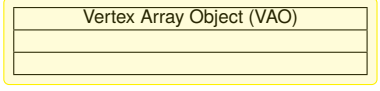
GPU

vertex shader

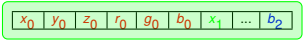
```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```



buffer0



CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,   0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

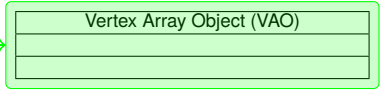
GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```



buffer0

x_0	y_0	z_0	r_0	g_0	b_0	x_1	...	b_2

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3,GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

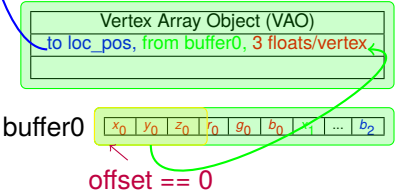
GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```



CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3,GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)
to loc_pos, from buffer0, 3 floats/vertex
to loc_col, from buffer1, 3 floats/vertex

buffer0

x_0	y_0	z_0	r_0	g_0	b_0	x_1	..	b_2

offset == 3x4

CPU

Interleaved attributes

```
# Setup the triangle with interleaved attributes
pos_col = np.array((( 0.0,  0.5, 0), (1, 0, 0),        # vertex 0
                    ( 0.5, -0.5, 0), (0, 1, 0),        # vertex 1
                    (-0.5, -0.5, 0), (0, 0, 1)), 'f') # vertex 2
stride = 6 * 4                      # 6 * sizeof(float)
offset_col = ctypes.c_void_p(3 * 4)     # cast as a pointer

self.buffers = [GL.glGenBuffers(1)]
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, self.buffers[0])
GL.glBufferData(GL.GL_ARRAY_BUFFER, pos_col, GL.GL_STATIC_DRAW)

self.glid = GL.glGenVertexArrays(1)
GL.glBindVertexArray(self.glid)
loc_pos = GL.glGetAttribLocation(shader.glid, "position")
GL.glEnableVertexAttribArray(loc_pos)
GL.glVertexAttribPointer(loc_pos, 3, GL.GL_FLOAT, False, stride, None)
loc_col = GL.glGetAttribLocation(shader.glid, "color")
GL.glEnableVertexAttribArray(loc_col)
GL.glVertexAttribPointer(loc_col, 3, GL.GL_FLOAT, False, stride, offset_col)

GL.glBindVertexArray(0)
GL.glBindBuffer(GL.GL_ARRAY_BUFFER, 0)
```

poscol

x_0	y_0	z_0	r_0	g_0	b_0	x_1	y_1	z_1	r_1	...	z_2	r_2	g_2	b_2

GPU

vertex shader

```
in vec3 position;
in vec3 color;
uniform mat4 matrix;
out vec3 fragColor;

void main() {
    gl_Position = matrix * vec4(position, 1);
    fragColor = color;
}
```

fragment shader

```
in vec3 fragColor;
out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1);
}
```

Vertex Array Object (VAO)
to loc_pos, from buffer0, 3 floats/vertex
to loc_col, from buffer1, 3 floats/vertex

buffer0

x_0	y_0	z_0	r_0	g_0	b_0	x_1	...	b_2