ModernGL Documentation

Release 4.0.0

Szabolcs Dombi

May 21, 2017

Contents:

1 ModernGL **1**
- 1.1 Context . . . 2
- 1.2 Shaders and Programs . . . 2
  - 1.2.1 Uniform . . . 2
  - 1.2.2 Uniform Blocks . . . 2
  - 1.2.3 Attributes . . . 2
  - 1.2.4 Varyings . . . 2
  - 1.2.5 Program Stages . . . 2
    - 1.2.5.1 Subroutines . . . 2
    - 1.2.5.2 SubroutineUniforms . . . 2
- 1.3 Buffers . . . 2
- 1.4 VertexArray . . . 2
  - 1.4.1 VertexArrayAttributes . . . 2
- 1.5 Textures . . . 2
- 1.6 Framebuffers . . . 2
- 1.7 Renderbuffers . . . 2
- 1.8 ComputeShaders . . . 2
- 1.9 InvalidObject . . . 2
- 1.10 Constants . . . 2
  - 1.10.1 Enable Flags . . . 2
  - 1.10.2 Versions . . . 2
  - 1.10.3 Primitives . . . 2
- 1.11 Error . . . 2

2 Examples **3**
- 2.1 01. Hello World! . . . 3
- 2.2 02. Uniforms and Attributes . . . 6
- 2.3 03. Blending . . . 7
- 2.4 04. Texture . . . 8
- 2.5 05. Perspective . . . 8
- 2.6 Julia Fractal . . . 8
- 2.7 Particle System . . . 11

3 Contributing **13**

4 Indices and tables **15**

CHAPTER 1

ModernGL

1.1 Context

1.2 Shaders and Programs

1.2.1 Uniform

1.2.2 Uniform Blocks

1.2.3 Attributes

1.2.4 Varyings

1.2.5 Program Stages

1.2.5.1 Subroutines

1.2.5.2 SubroutineUniforms

1.3 Buffers

1.4 VertexArray

1.4.1 VertexArrayAttributes

1.5 Textures

1.6 Framebuffers

1.7 Renderbuffers

1.8 ComputeShaders

CHAPTER 2

Examples

2.1 01. Hello World!

Fig. 2.1: Hello World!

```
import struct

import GLWindow
import ModernGL

# Window & Context

wnd = GLWindow.create_window()
ctx = ModernGL.create_context()

# Shaders & Program

prog = ctx.program([
        ctx.vertex_shader('''
                #version 330

                in vec2 vert;

                void main() {
                        gl_Position = vec4(vert, 0.0, 1.0);
                }
        '''),
        ctx.fragment_shader('''
                #version 330

                out vec4 color;

                void main() {
                        color = vec4(0.3, 0.5, 1.0, 1.0);
                }
        '''),
])

# Buffer

vbo = ctx.buffer(struct.pack('6f',
        0.0, 0.8,
        -0.6, -0.8,
        0.6, -0.8,
))

# Put everything together

vao = ctx.simple_vertex_array(prog, vbo, ['vert'])

# Main loop

while wnd.update():
        ctx.viewport = wnd.viewport
        ctx.clear(240, 240, 240)
        vao.render()
```

Fig. 2.2: Uniforms and Attributes

2.2 02. Uniforms and Attributes

```
import struct

import GLWindow
import ModernGL

# Window & Context

wnd = GLWindow.create_window()
ctx = ModernGL.create_context()

# Shaders & Program

prog = ctx.program([
        ctx.vertex_shader('''
                #version 330

                in vec2 vert;

                in vec3 vert_color;
                out vec3 frag_color;

                uniform vec2 scale;
                uniform float rotation;

                void main() {
                        frag_color = vert_color;
                        mat2 rot = mat2(
                                cos(rotation), sin(rotation),
                                -sin(rotation), cos(rotation)
                        );
                        gl_Position = vec4((rot * vert) * scale, 0.0, 1.0);
                }
        '''),
        ctx.fragment_shader('''
                #version 330

                in vec3 frag_color;
                out vec4 color;

                void main() {
                        color = vec4(frag_color, 1.0);
                }
        '''),
])

# Uniforms

scale = prog.uniforms['scale']
rotation = prog.uniforms['rotation']

width, height = wnd.size
scale.value = (height / width * 0.75, 0.75)

# Buffer

vbo = ctx.buffer(struct.pack('15f',
```

```
        1.0, 0.0,
        1.0, 0.0, 0.0,

        -0.5, 0.86,
        0.0, 1.0, 0.0,

        -0.5, -0.86,
        0.0, 0.0, 1.0,
))

# Put everything together

vao = ctx.simple_vertex_array(prog, vbo, ['vert', 'vert_color'])

# Main loop

while wnd.update():
        ctx.viewport = wnd.viewport
        ctx.clear(240, 240, 240)
        rotation.value = wnd.time
        vao.render()
```

2.3 03. Blending

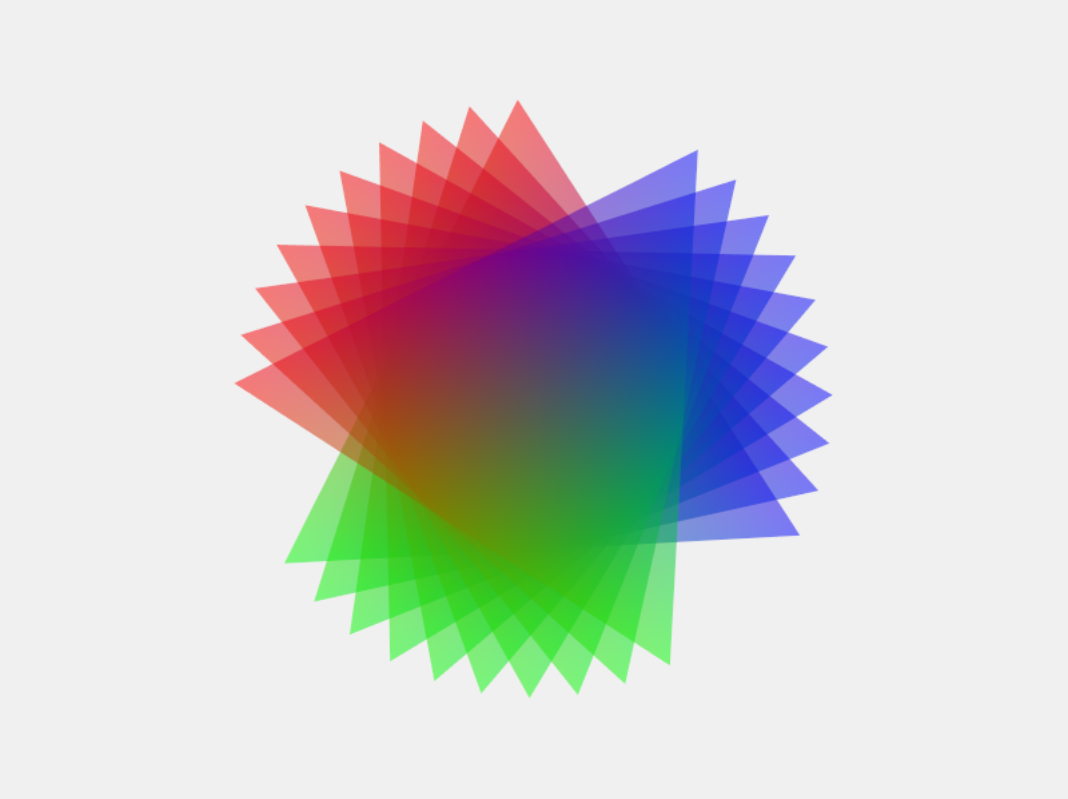

Fig. 2.3: Blending

2.4 04. Texture

Fig. 2.4: Texture

2.5 05. Perspective

2.6 Julia Fractal

```
import struct

import GLWindow
import ModernGL

# Window & Context

wnd = GLWindow.create_window()
ctx = ModernGL.create_context()

vert = ctx.vertex_shader('''
        #version 330

        in vec2 vert;
        out vec2 tex;

        void main() {
```

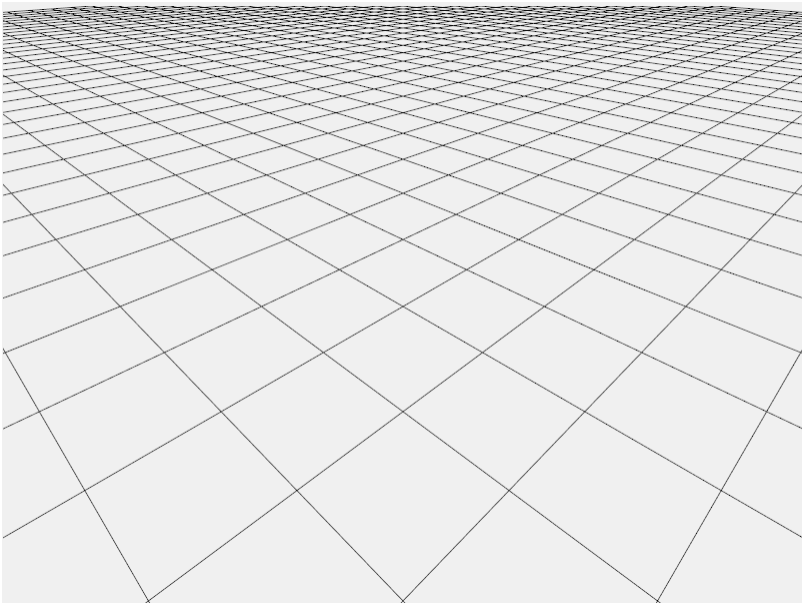

Fig. 2.5: Perspective projection

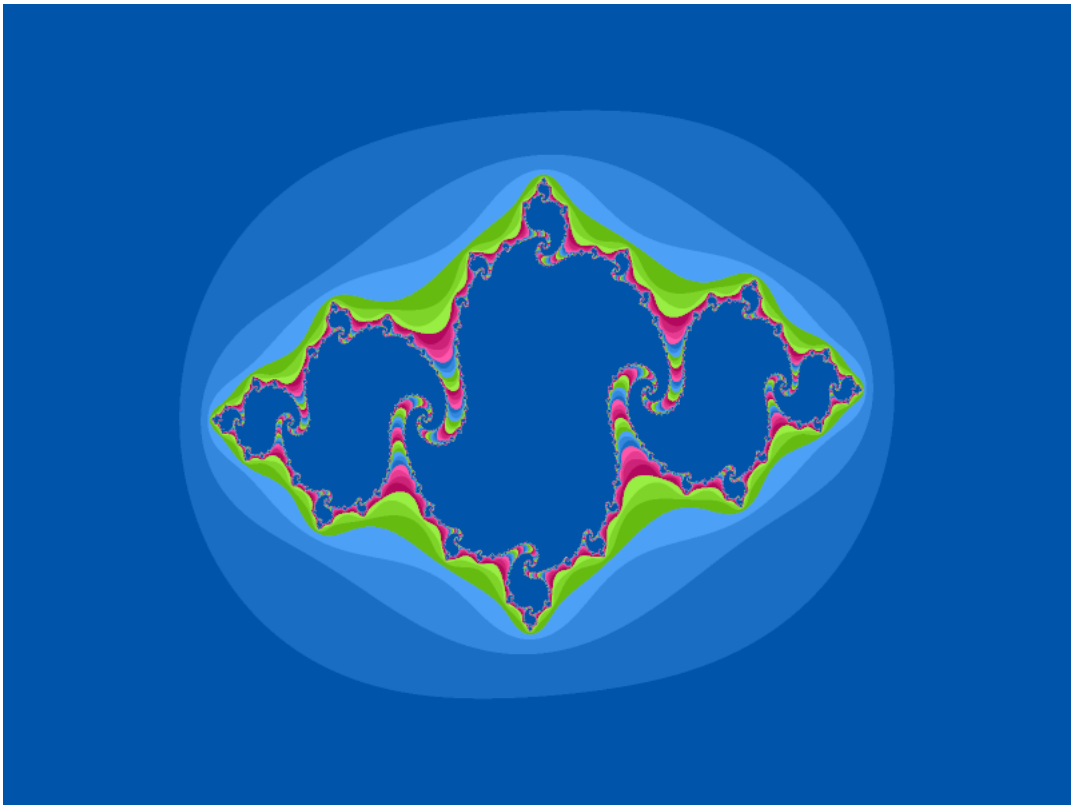

Fig. 2.6: Julia Fractal

```
                gl_Position = vec4(vert, 0.0, 1.0);
                tex = vert / 2.0 + vec2(0.5, 0.5);
        }
''')

frag = ctx.fragment_shader('''
        #version 330

        in vec2 tex;
        out vec4 color;

        uniform vec2 center;
        uniform int iter;

        void main() {
                vec2 z = vec2(5.0 * (tex.x - 0.5), 3.0 * (tex.y - 0.5));
                vec2 c = center;

                int i;
                for(i = 0; i < iter; i++) {
                        vec2 v = vec2(
                                (z.x * z.x - z.y * z.y) + c.x,
                                (z.y * z.x + z.x * z.y) + c.y
                        );
                        if (dot(v, v) > 4.0) break;
                        z = v;
                }

                float cm = fract((i == iter ? 0.0 : float(i)) * 10 / iter);
                color = vec4(
                        fract(cm + 0.0 / 3.0),
                        fract(cm + 1.0 / 3.0),
                        fract(cm + 2.0 / 3.0),
                        1.0
                );
        }
''')

prog = ctx.program([vert, frag])

vbo = ctx.buffer(struct.pack('8f', -1.0, -1.0, -1.0, 1.0, 1.0, -1.0, 1.0, 1.0))
vao = ctx.simple_vertex_array(prog, vbo, ['vert'])

prog.uniforms['iter'].value = 100

x, y = (0.49, 0.32)

wnd.grab_mouse(True)

while wnd.update():
        ctx.viewport = wnd.viewport
        ctx.clear(240, 240, 240)
        mx, my = wnd.mouse_delta
        x -= mx / 100
        y -= my / 100

        prog.uniforms['center'].value = (y, x)
        vao.render(ModernGL.TRIANGLE_STRIP)
```

2.7 Particle System

Fig. 2.7: Particle System

```
import math
import random
import struct

import GLWindow
import ModernGL

# Window & Context

wnd = GLWindow.create_window()
ctx = ModernGL.create_context()

tvert = ctx.vertex_shader('''
        #version 330

        uniform vec2 acc;

        in vec2 in_pos;
        in vec2 in_prev;

        out vec2 out_pos;
        out vec2 out_prev;

        void main() {
                out_pos = in_pos * 2.0 - in_prev + acc;
                out_prev = in_pos;
        }
''')

vert = ctx.vertex_shader('''
        #version 330
```

```
32
33          in vec2 vert;
34
35          void main() {
36                  gl_Position = vec4(vert, 0.0, 1.0);
37          }
38  ''')
39
40  frag = ctx.fragment_shader('''
41          #version 330
42
43          out vec4 color;
44
45          void main() {
46                  color = vec4(0.30, 0.50, 1.00, 1.0);
47          }
48  ''')
49
50  prog = ctx.program([vert, frag])
51
52  transform = ctx.program(tvert, ['out_pos', 'out_prev'])
53
54  def particle():
55          a = random.uniform(0.0, math.pi * 2.0)
56          r = random.uniform(0.0, 0.001)
57
58          return struct.pack('2f2f', 0.0, 0.0, math.cos(a) * r - 0.003, math.sin(a) * r␣
    ↪- 0.008)
59
60  vbo1 = ctx.buffer(b''.join(particle() for i in range(1024)))
61  vbo2 = ctx.buffer(reserve = vbo1.size)
62
63  vao1 = ctx.simple_vertex_array(transform, vbo1, ['in_pos', 'in_prev'])
64  vao2 = ctx.simple_vertex_array(transform, vbo2, ['in_pos', 'in_prev'])
65
66  render_vao = ctx.vertex_array(prog, [
67          (vbo1, '2f8x', ['vert']),
68  ])
69
70  transform.uniforms['acc'].value = (0, -0.0001)
71
72  idx = 0
73
74  ctx.point_size = 5.0
75
76  while wnd.update():
77          ctx.viewport = wnd.viewport
78          ctx.clear(240, 240, 240)
79
80          for i in range(8):
81                  vbo1.write(particle(), offset = idx * struct.calcsize('2f2f'))
82                  idx = (idx + 1) % 1024
83
84          render_vao.render(ModernGL.POINTS, 1024)
85          vao1.transform(vbo2, ModernGL.POINTS, 1024)
86          ctx.copy_buffer(vbo1, vbo2)
```

CHAPTER 3

Contributing

CHAPTER 4

Indices and tables

- genindex
- modindex
- search