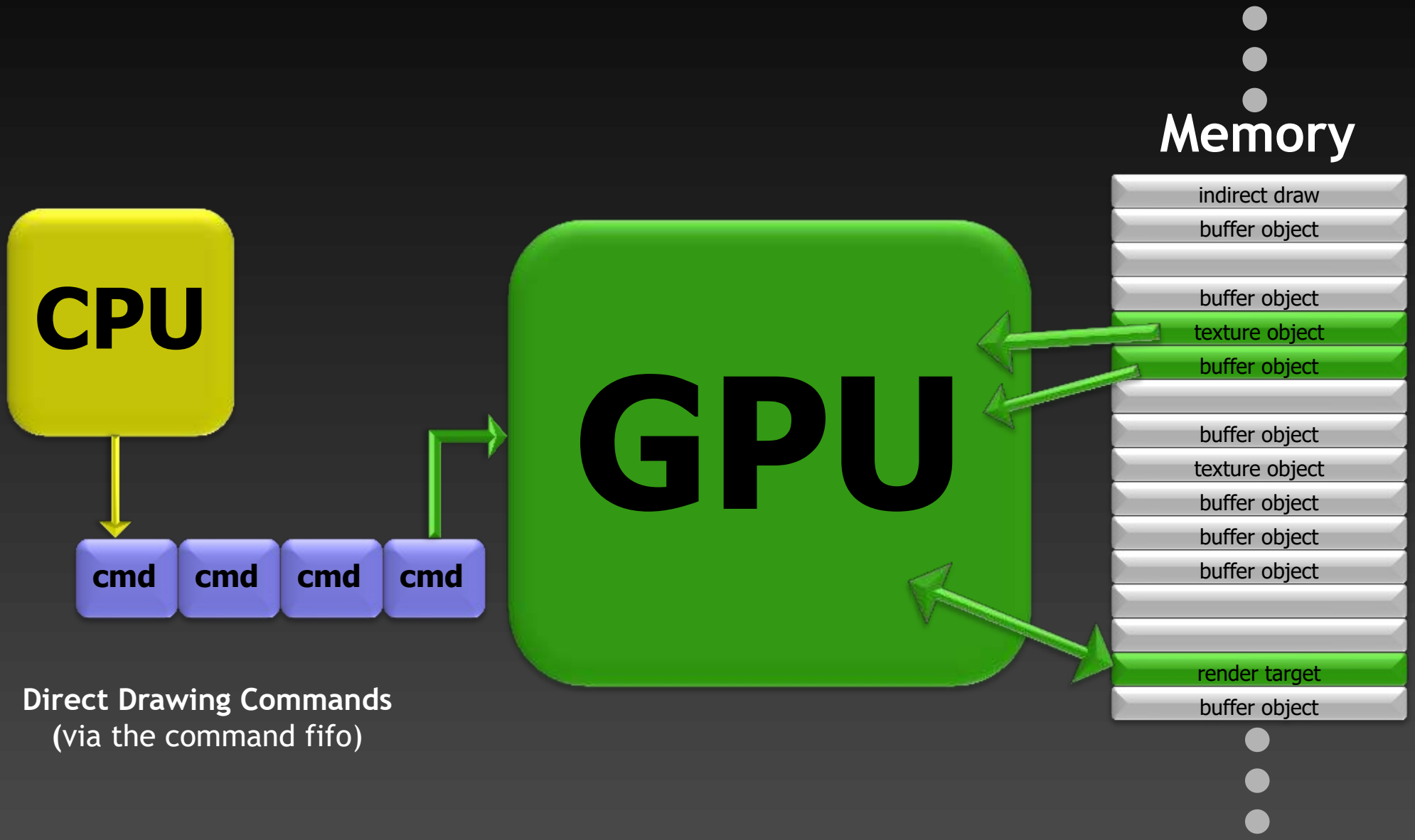
Modern OpenGL
DSA - Direct State Access

- OpenGL = state machine
  - to modify data, we need to bind it first
    - bind(res_A), modify_prop_foo(), modify_prop_bar(),
    - bind(res_B), modify_prop_foo(),…
  - slower (more calls)
  - error prone (modifying different object by accident)
  - prevents multithreading
- Explicit binding is expensive → get rid of it!
  - bindless acces = Direct State Access = MODIFY without binding
- AZDO → Approaching Zero Driver Overhead
  - reduce API calls to minimum
- Support
  - OpenGL and shaders – Core, version **4.5+**
    - first line of the shader must be (at least)

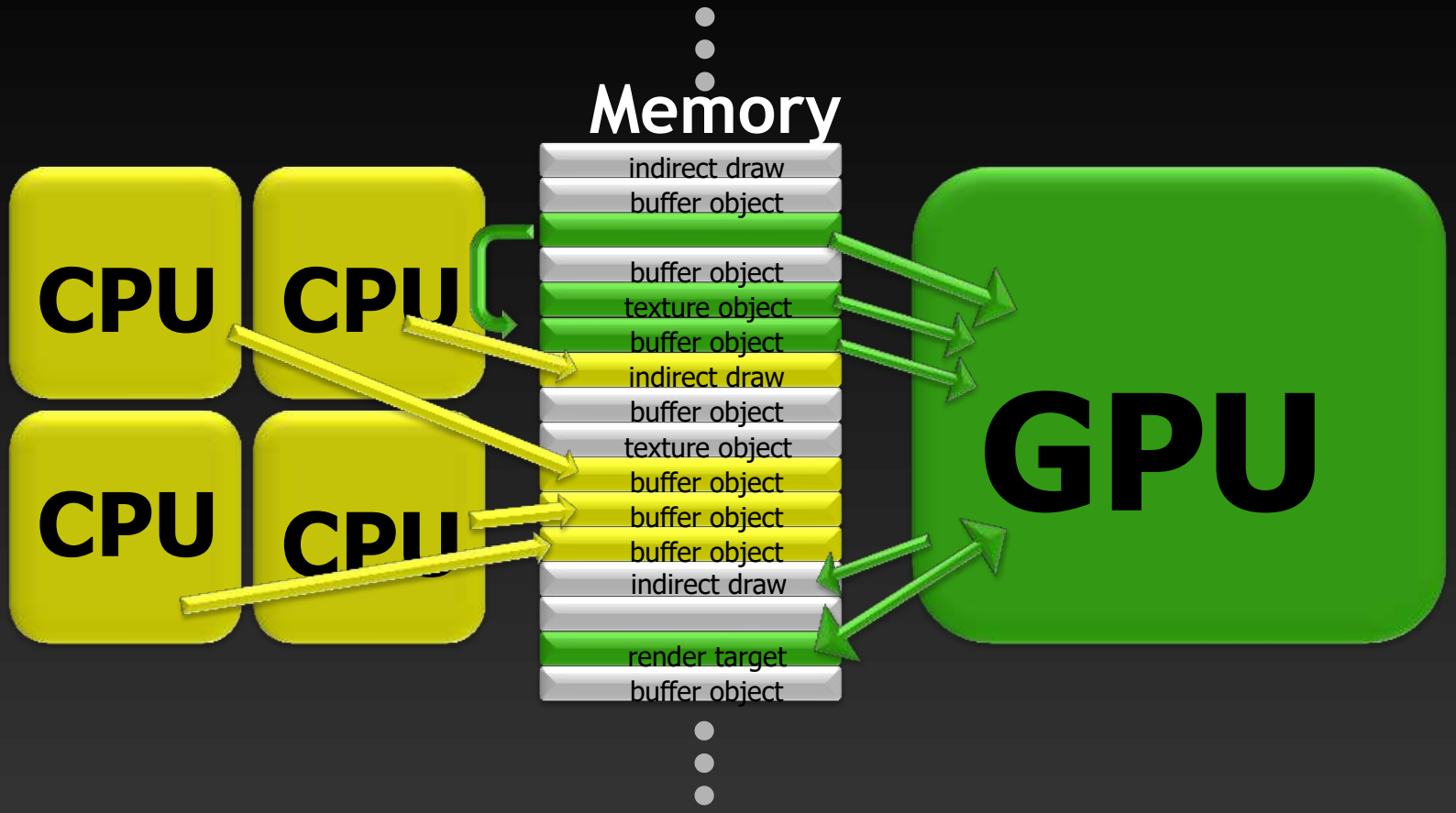
```
#version 450 core
```

```
if (!GLEW_ARB_direct_state_access)
     throw std::runtime_error("No DSA :-(");
```

Classic OpenGL Model



Efficient OpenGL Model



- CPU and GPU decoupled
- CPU writes to memory – multithreaded (no API calls!)
- GPU writes to memory – still no API
- GPU reads from memory – minimal API
- SPEEDUP 5x-15x

DSA Nomenclature

OpenGL Object Type	Context Object Name	DSA Object Name
Texture Object	"Tex"	"Texture"
Framebuffer Object	„Framebuffer"	"NamedFramebuffer"
Buffer Object	"Buffer"	"NamedBuffer"
Transform Feedback Object	"TransformFeedback"	"TransformFeedback"
Vertex Array Object	N/A	"VertexArray"
Sampler Object	N/A	"Sampler"
Query Object	N/A	"Query"
Program Object	N/A	"Program"

- Syntax
  - Verb-Object-Command syntax
  - so-called „named objects“
    - sadly, inconsistent OpenGL naming convention

Buffer Objects

- glGenBuffers + glBindBuffer → glCreateBuffers
- glBufferData → glNamedBufferData
- glVertexAttribPointer → glVertexAttribFormat + glVertexAttribBinding
- Let's have:

```
struct vertex {
glm::vec3 pos;
glm::vec2 texcoord;
};

std::vector<vertex> vertices;
std::vector<GLuint> indices;
```

Buffers: non-DSA → DSA

```
GLuint VAO, VBO, EBO;
glGenVertexArrays(1, &VAO);
glGenBuffers(1, &VBO);
glGenBuffers(1, &EBO);

glBindVertexArray(VAO);

glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, vertices.size(), vertices.data(), GL_STATIC_DRAW);

glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, indices.size(), indices.data(), GL_STATIC_DRAW);

glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(vertex), (void*)offsetof(vertex,pos));
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, sizeof(vertex), (void*)offsetof(vertex, pos));

glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
```

- DSA – no binding required for VAO, buffers; added attr binding

```
GLuint VAO, VBO, EBO;
glCreateVertexArrays(1, &VAO);
glCreateBuffers(1, &VBO);
glCreateBuffers(1, &EBO);

glNamedBufferData(VBO, vertices.size(), vertices.data(), GL_STATIC_DRAW);
glNamedBufferData(EBO, indices.size(), indices.data(), GL_STATIC_DRAW);

glEnableVertexArrayAttrib(VAO, 0);
glVertexArrayAttribBinding(VAO, 0, 0);
glVertexArrayAttribFormat(VAO, 0, 3, GL_FLOAT, GL_FALSE, offsetof(vertex, pos));

glEnableVertexArrayAttrib(VAO, 1);
glVertexArrayAttribBinding(VAO, 1, 0);
glVertexArrayAttribFormat(VAO, 1, 2, GL_FLOAT, GL_FALSE, offsetof(vertex,texcoord));

glVertexArrayVertexBuffer(VAO, 0, VBO, 0, sizeof(vertex));
glVertexArrayElementBuffer(VAO, EBO);
```

DSA: draw (VAO use)

- non-DSA & DSA (same)
  - use with binding

```
glUseProgram(shaderProgram);
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_INT, 0);
```

DSA: textures

- non-DSA

```
GLuint tex;
glGenTextures(1, &tex); // create
glActiveTexture(0);
glBindTexture(GL_TEXTURE_2D, tex); // initialize

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, widthImg, heightImg, 0, GL_RGBA, GL_UNSIGNED_BYTE, byteptr); // mutable texture
glGenerateMipmap(GL_TEXTURE_2D);

glBindTexture(GL_TEXTURE_2D, 0);
```

- DSA

```
GLuint tex;
glCreateTextures(GL_TEXTURE_2D, 1, &tex); // create AND initialize

glTextureParameteri(tex, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTextureParameteri(tex, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTextureParameteri(tex, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTextureParameteri(tex, GL_TEXTURE_WRAP_T, GL_REPEAT);

// no TexImage*D, immutable texture objects only (safer, faster)
glTextureStorage2D(tex, 1, GL_RGBA8, widthImg, heightImg); // create empty texture object
glTextureSubImage2D(tex, 0, 0, 0, widthImg, heightImg, GL_RGBA, GL_UNSIGNED_BYTE, byteptr); //set data (params: MIP lvl 0, x-offs, y-
offs)
glGenerateTextureMipmap(tex); // generate MIPMAP
```

DSA: texture use

- non-DSA

```
glUseProgram(shaderProgram);
glActiveTexture(0);
glBindTexture(GL_TEXTURE_2D, tex);
glUniform1i(glGetUniformLocation(shaderProgram, "tex"), 0);
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_INT, 0);
```

- DSA: must bind, but activate and bind together

```
glUseProgram(shaderProgram);
glBindTextureUnit(0, tex);
glUniform1i(glGetUniformLocation(shaderProgram, "tex"), 0);
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, indices.size(), GL_UNSIGNED_INT, 0);
```

DSA: framebuffer objects

- non-DSA

```
GLuint FBO;
glGenFramebuffers(1, &FBO);
glBindFramebuffer(GL_FRAMEBUFFER, FBO);

GLuint framebufferTex;
glGenTextures(1, &framebufferTex);
glActiveTexture(0);
glBindTexture(GL_TEXTURE_2D, framebufferTex);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, SCREEN_WIDTH, SCREEN_HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, NULL);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, framebufferTex, 0);
glBindTexture(GL_TEXTURE_2D, 0);

auto fboStatus = glCheckFramebufferStatus(GL_FRAMEBUFFER);
if (fboStatus != GL_FRAMEBUFFER_COMPLETE)
std::cout << "Framebuffer error: " << fboStatus << "\n";
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

- DSA

```
GLuint FBO;
glCreateFramebuffers(1, &FBO);

GLuint framebufferTex;
glCreateTextures(GL_TEXTURE_2D, 1, &framebufferTex); // create AND initialize
glTextureParameteri(framebufferTex, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTextureParameteri(framebufferTex, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTextureParameteri(framebufferTex, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTextureParameteri(framebufferTex, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTextureStorage2D(framebufferTex, 1, GL_RGB8, SCREEN_WIDTH, SCREEN_HEIGHT);
glNamedFramebufferTexture(FBO, GL_COLOR_ATTACHMENT0, framebufferTex, 0);
// or glNamedFramebufferTexture(fbo, GL_DEPTH_ATTACHMENT, depthTex, 0) etc...

auto fboStatus = glCheckNamedFramebufferStatus(FBO, GL_FRAMEBUFFER);
if (fboStatus != GL_FRAMEBUFFER_COMPLETE)
std::cout << "Framebuffer error: " << fboStatus << "\n";
```

DSA: FBO use

- non-DSA & DSA (same)
  - use with binding

```
glBindFramebuffer(GL_FRAMEBUFFER, FBO);
GLfloat backgroundColor[] = { 19.0f / 255.0f, 34.0f / 255.0f, 44.0f / 255.0f, 1.0f };
glClearNamedFramebufferfv(FBO, GL_COLOR, 0, backgroundColor);  // colorbuff index
```

- Blitting

```
glBindFramebuffer(GL_READ_FRAMEBUFFER, fbo_src);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, fbo_dst);

glBlitFramebuffer(src_x, src_y, src_w, src_h, dst_x, dst_y, dst_w, dst_h, GL_COLOR_BUFFER_BIT, GL_LINEAR);
```

- Becomes

```
glBlitNamedFramebuffer(fbo_src, fbo_dst, src_x, src_y, src_w, src_h, dst_x, dst_y, dst_w, dst_h, GL_COLOR_BUFFER_BIT, GL_LINEAR);
```

Clearing Buffers

- Non-DSA
- usual way

```
glBindFramebuffer(GL_FRAMEBUFFER, fb);
glClearColor(r, g, b, a);
glClearDepth(d);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

- more versatile, per-attachement

```
glBindFramebuffer(GL_FRAMEBUFFER, fb);
glClearBufferfv(GL_COLOR, GL_DRAW_BUFFER0 + col_buff_index, &rgba);
glClearBufferfv(GL_DEPTH, 0, &d);
```

- DSA

```
glClearNamedFramebufferfv(fb, GL_COLOR, col_buff_index, &rgba);
glClearNamedFramebufferfv(fb, GL_DEPTH, 0, &d);
```