OpenGL Buffer Objects

- Vertex Buffer Object (VBO)
- Frame Buffer Object (FBO)
- Pixel Buffer Object (PBO)

Vertex Buffer Objects

- Geometry better accessible by GPU
  - Vertex Buffer
    - Geometry – position, normal, color, texcoord, general attributes...
  - Index Buffer
    - Vertex IDs for primitive assembly
      - GL_TRIANGLES – 3 IDs per primitive
      - GL_QUADS – 4 IDs per primitive
      - ...

GL Buffers – create & destroy

- void glGenBuffers{ARB} (GLsizei n, GLuint * bufs)
  - generate buffer object “names” (IDs)
  - n – number of buffers to be created
  - bufs – array of size n for new buffer IDs
- void glDeleteBuffers{ARB} (GLsizei n, const GLuint * bufs)
  - delete named buffer objects
  - n – number of buffers to be deleted
  - bufs – array of size n for buffer “names” (IDs) to be deleted

glBufferData

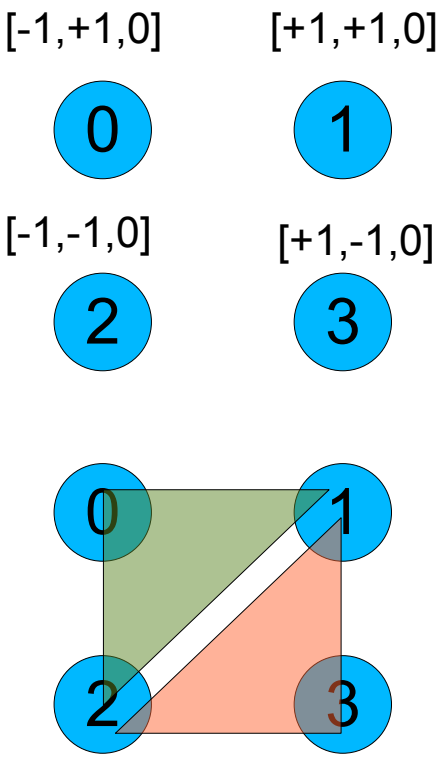
- creates and initializes a buffer object's data store
- void glBufferData{ARB} (GLenum target, GLsizeiptr size,
  const GLvoid *data, GLenum usage)
  - target
    - ARRAY, ELEMENT_ARRAY,
    - PIXEL_PACK, PIXEL_UNPACK
  - size – number of bytes
  - data – host memory block to be copied into glBuffer
    - NULL – no copying, just allocating
  - usage
    - GL_STREAM_DRAW, _READ, _COPY
    - GL_STATIC_DRAW, _READ, _COPY,
    - GL_DYNAMIC_DRAW, _READ, _COPY

glBindBuffer

- void glBindBuffer{ARB} (GLenum target, GLuint bufID)
  - target
    - GL_ARRAY_BUFFER – vertex buffer
    - GL_ELEMENT_ARRAY_BUFFER – index buffer
    - GL_PIXEL_PACK_BUFFER – pixel buffer
    - GL_PIXEL_UNPACK_BUFFER – pixel buffer
  - bufID
    - Generated using glGenBuffers{ARB}
    - ID==0 – unbind buffer

Geometry representation in system memory

- Data in system memory
  - float gPositions [ ] = {
  - -1,+1, 0, +1,+1, 0,
  - -1,-1, 0, -1,+1, 0
  - };
  - unsigned int gVertexIDs [ ] ={
    - 0, 1, 2,
    - 2, 1, 3
  - };

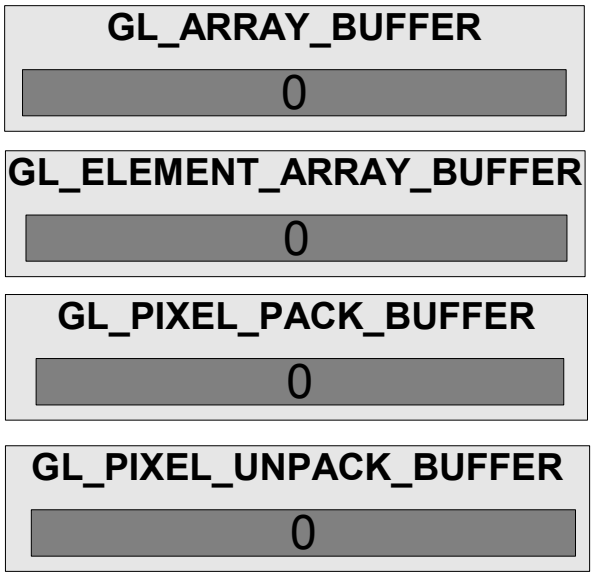


Init VBOs and transfer from system memory

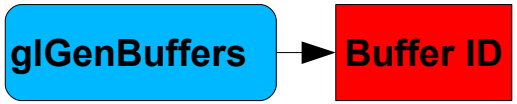
- VertexBuffer VBO
  - glGenBuffers(1, &gPositionsVB);
  - glBindBuffer(GL_ARRAY_BUFFER, gPositionsVB);
  - glBufferData(GL_ARRAY_BUFFER, sizeof(gPositions), gPositions,GL_STATIC_DRAW_ARB);
- Index Buffer VBO
  - glGenBuffers(1, &gTrianglesIB); // Index Buffer
  - glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,gTrianglesIB);
  - glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(gVertexIDs), gVertexIDs,GL_STATIC_DRAW_ARB);

glBuffers
data in GPU mem

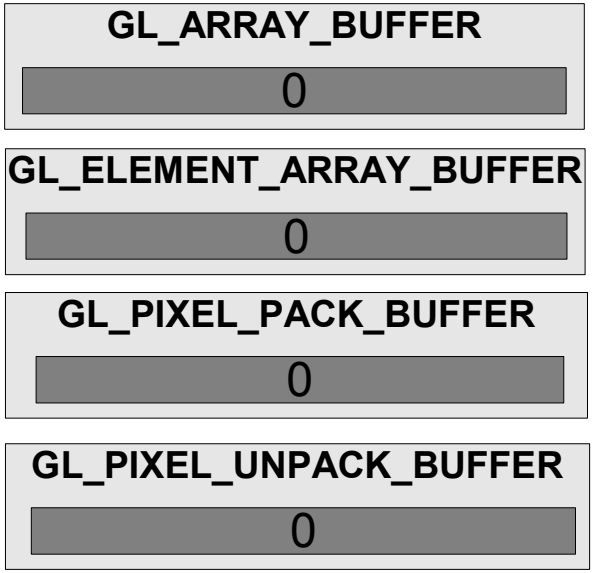
- 4 types of glBuffers
  - GL_ARRAY_BUFFER
    - attribute array buffer (VBO)
    - positions, normals, texcoords, attributes, ...
  - GL_ELEMENT_ARRAY_BUFFER
    - index (element) array buffer
  - GL_PIXEL_PACK_BUFFER
    - pixel buffer (PBO)
  - GL_PIXEL_UNPACK_BUFFER
    - pixel buffer (PBO)



glBuffers
data in GPU mem



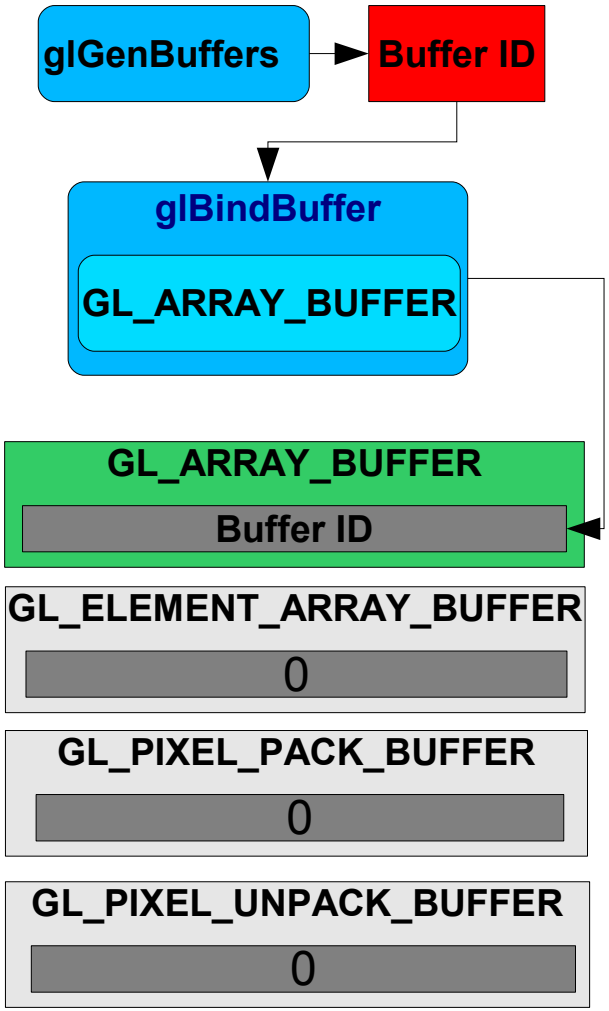
- How to create it ? Generate ID !
- void **glGenBuffersARB** (GLsizei n, GLuint * bufs)
  - generate buffer object IDs
  - n – number of buffers to be created
  - bufs – array of size n for new buffer IDs



glBuffers
data in GPU mem

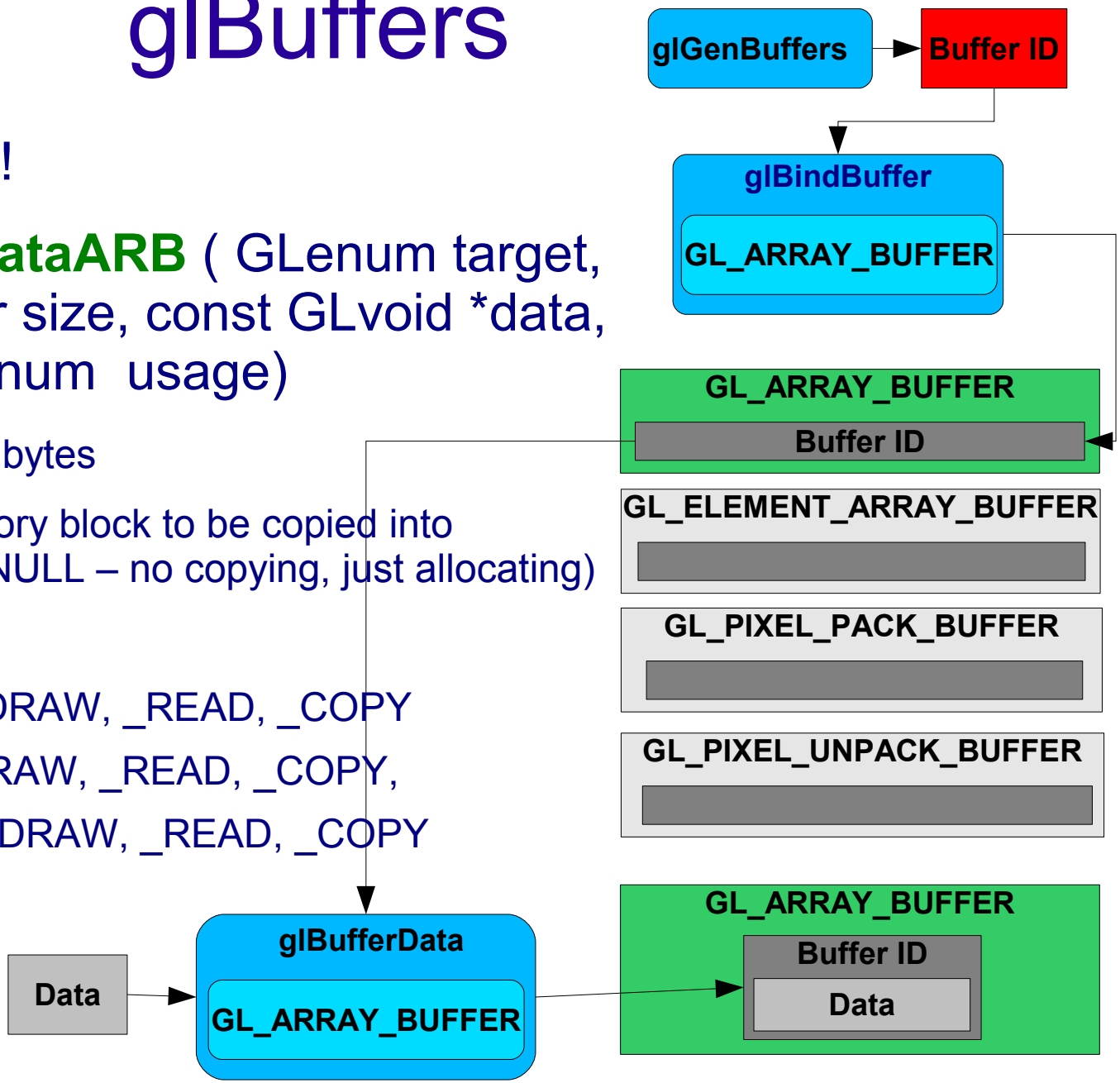
- How to use it ? Bind it !
- void **glBindBufferARB** (GLenum target,

  GLuint bufID)
  - target
    - GL_ARRAY_BUFFER
    - GL_ELEMENT_ARRAY_BUFFER
    - GL_PIXEL_PACK_BUFFER
    - GL_PIXEL_UNPACK_BUFFER
      - bufID
        - Generated using **glGenBuffers**
        - ID==0 – unbind buffer



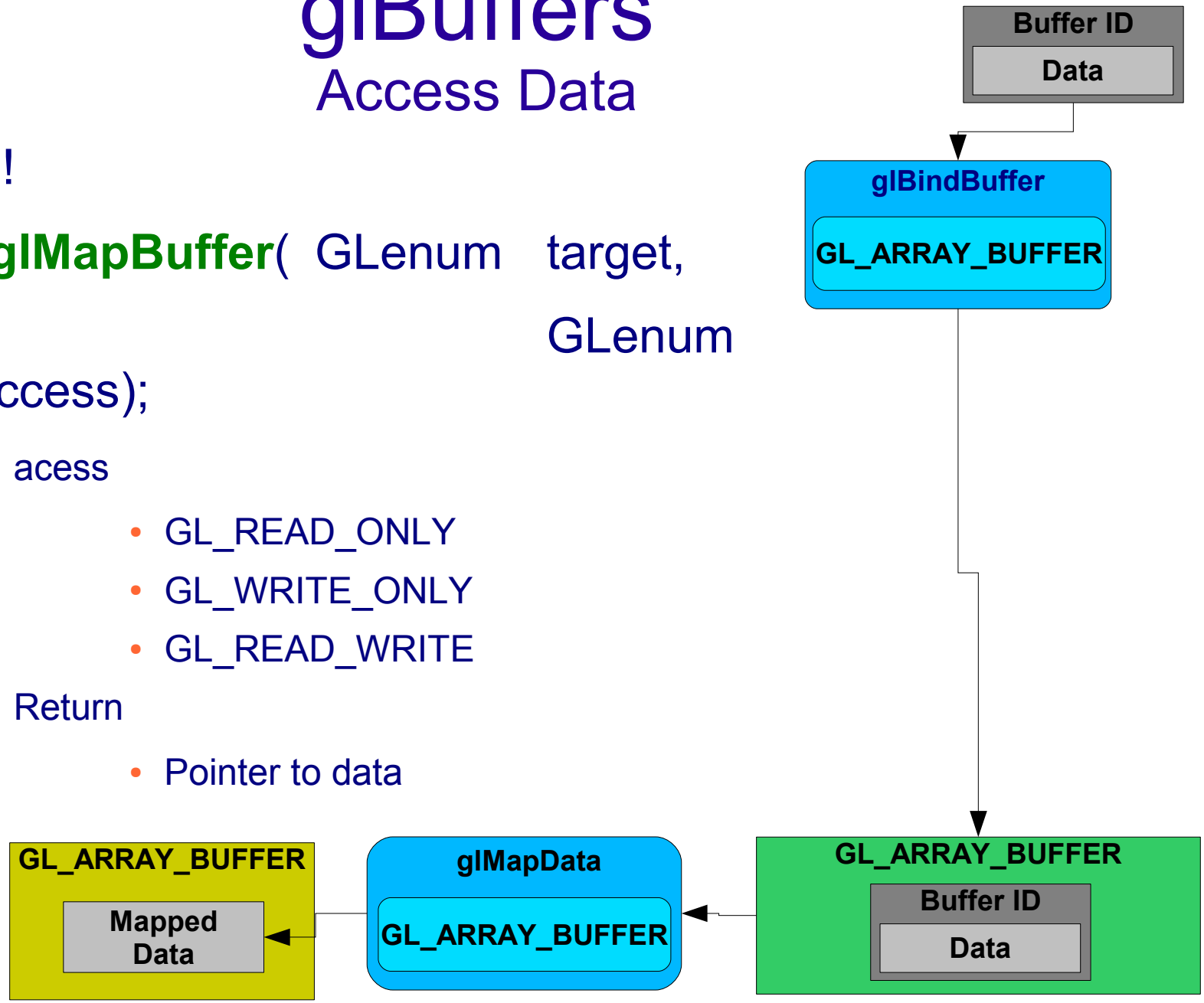
glBuffers

- Allocate / Fill it !
- void **glBufferDataARB** (GLenum target,
  GLsizeiptr size, const GLvoid *data,
  GLenum usage)
  - size – number of bytes
  - data – host memory block to be copied into glBuffer (NULL – no copying, just allocating)
  - usage
    - GL_STREAM_DRAW, _READ, _COPY
    - GL_STATIC_DRAW, _READ, _COPY,
    - GL_DYNAMIC_DRAW, _READ, _COPY



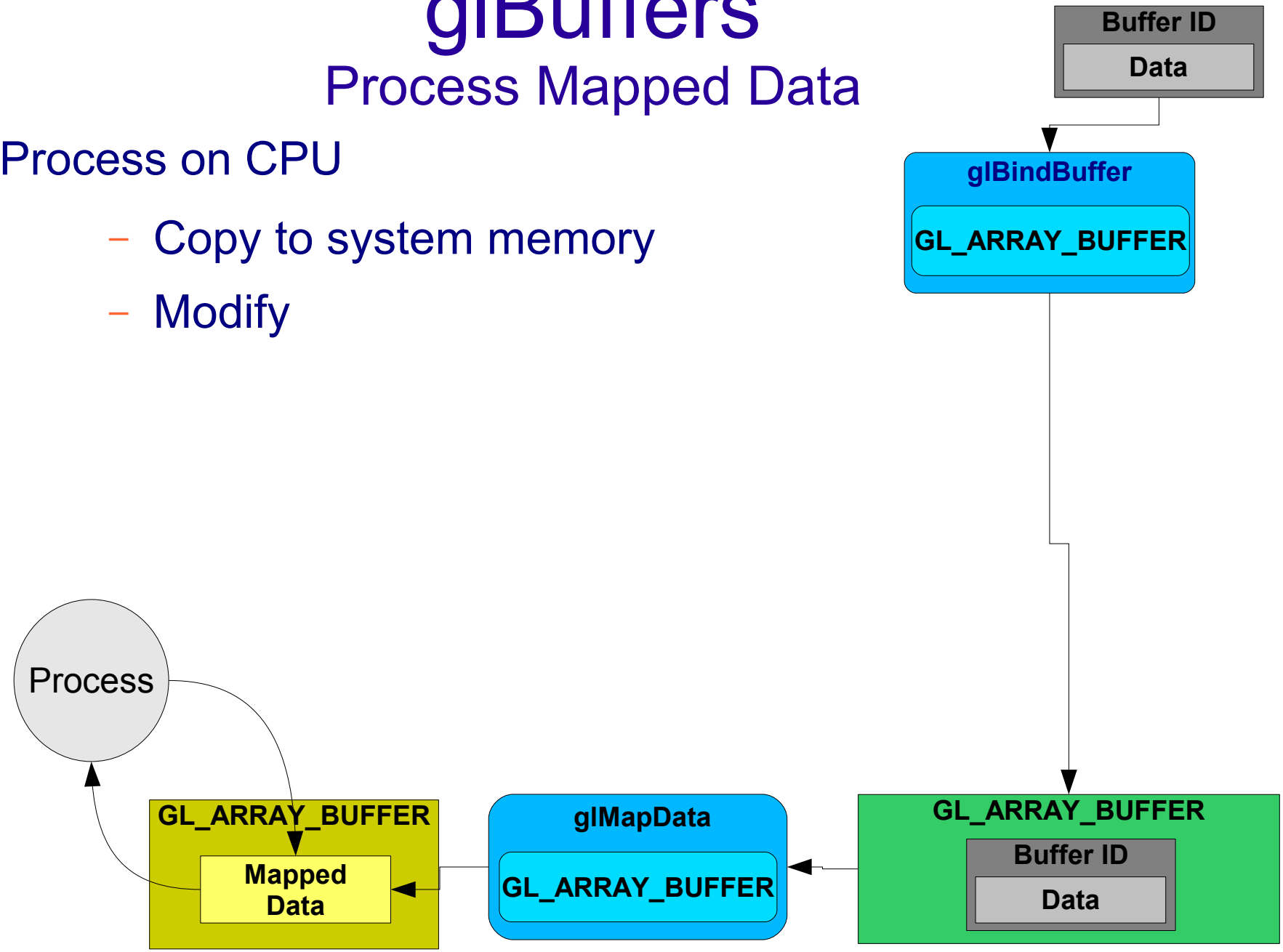
glBuffers
Access Data

- Map It !
- void * **glMapBuffer**(GLenum target, GLenum access);
  - acess
    - GL_READ_ONLY
    - GL_WRITE_ONLY
    - GL_READ_WRITE
  - Return
    - Pointer to data



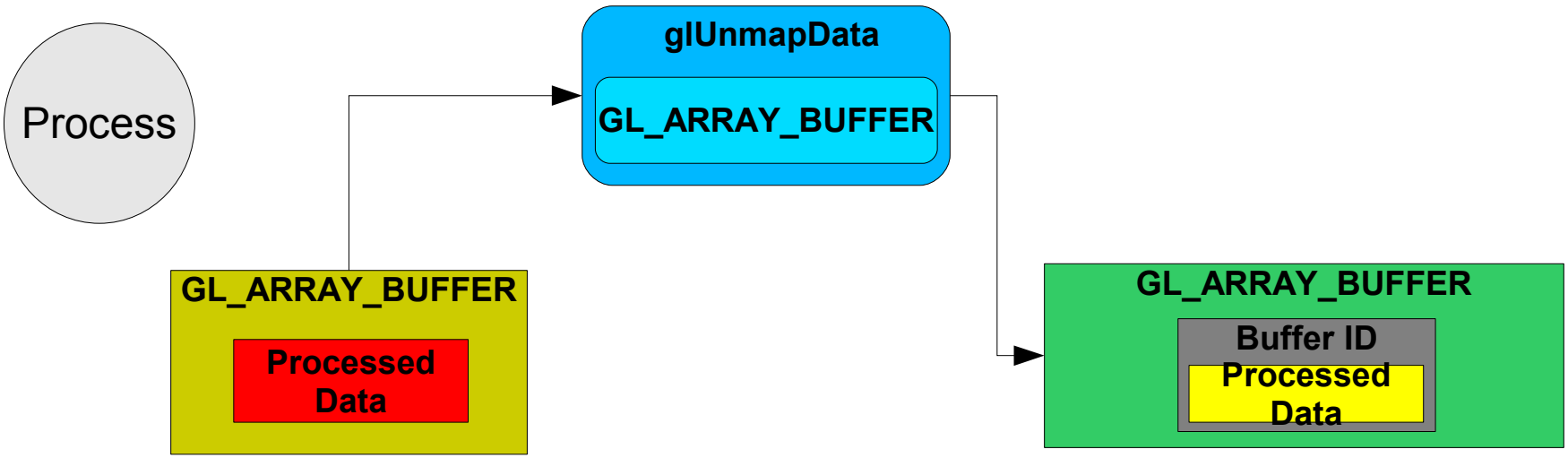
glBuffers
Process Mapped Data

- Process on CPU
  - Copy to system memory
  - Modify



glBuffers
Unmap Data

- GLboolean **glUnmapBuffer**(GLenum target)

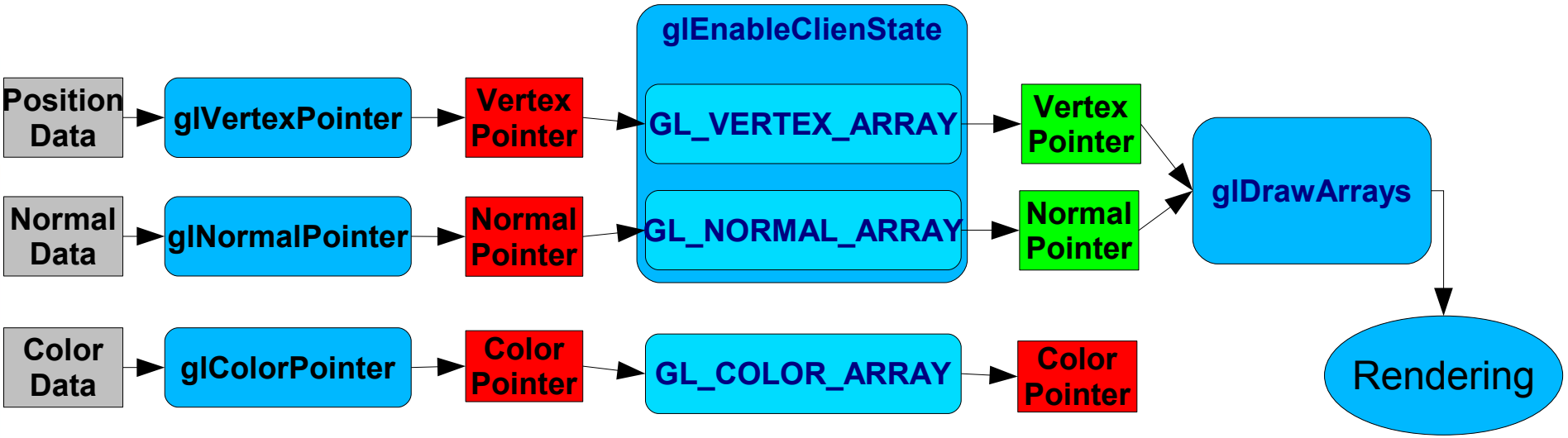


Rendering with VBO

- glBindBufferARB(GL_ARRAY_BUFFER_ARB, gPositionsVB);
- // set Vertex Pointer to the bound array buffer
- glVertexPointer(3, GL_FLOAT, 0, (char *) NULL);
- // bind index buffer
- glBindBufferARB(GL_ELEMENT_ARRAY_BUFFER_ARB, gTrianglesIB);
- // use only vertex positions (no normals, texcoords, colors,...)
- glEnableClientState(GL_VERTEX_ARRAY);
- // draw primitives using bound element array buffer
- glDrawElements(GL_TRIANGLES, 2*3, GL_UNSIGNED_INT, NULL);
  - // draw primitives using bound vertex buffers
  - glDrawArrays(GL_QUADS, 0, 1*4);
  - // draw using bound element array with limited index range
  - glDrawRangeElements(GL_TRIANGLES, 0, 3, 2*3, GL_UNSIGNED_INT, NULL);

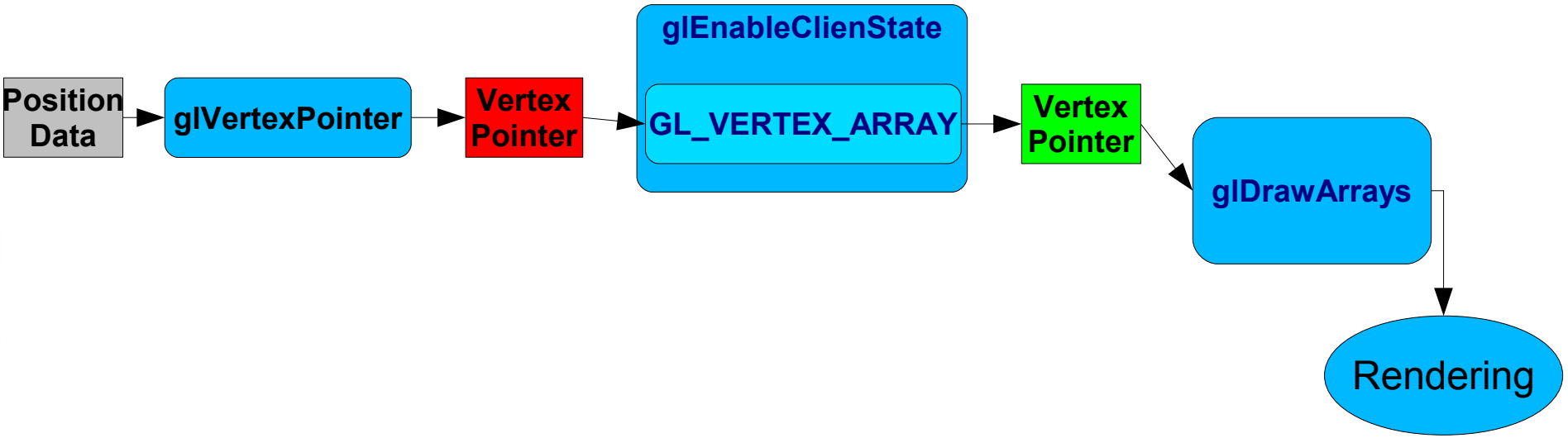
glDrawArrays
rendering data from CPU mem

- void **glDrawArrays**(GLenum mode, GLint first, GLsizei count)
  - mode - GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, GL_POLYGON
  - first - Specifies the starting index in the enabled arrays.
  - count - Specifies the number of indices to be rendered.



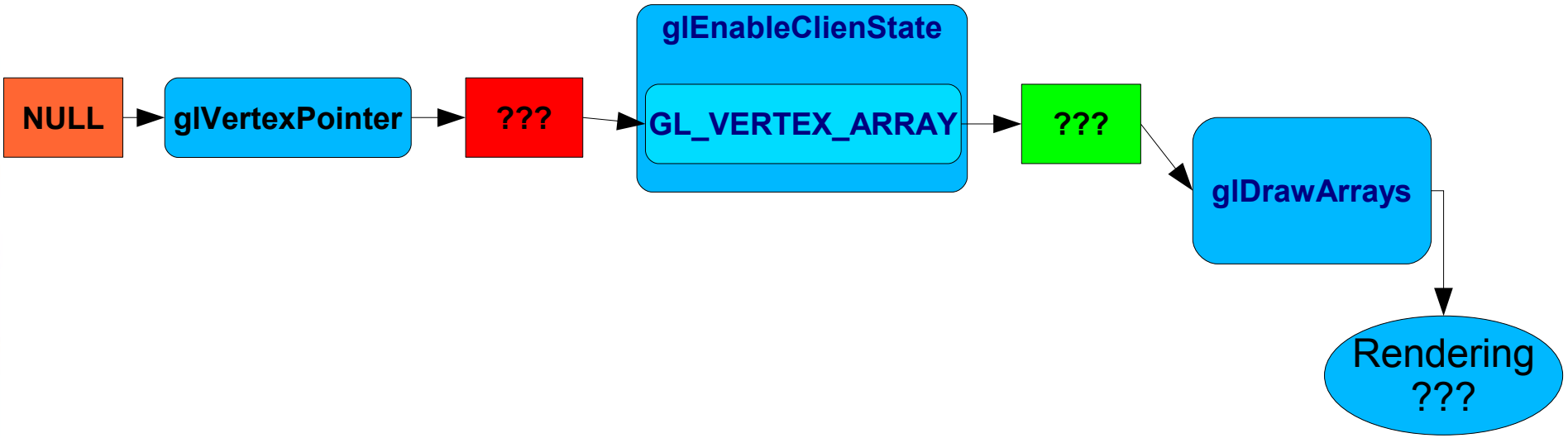
glDrawArrays
rendering data from CPU mem

- void **glDrawArrays**(GLenum mode, GLint first, GLsizei count)
  - mode - GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, GL_POLYGON
  - first - Specifies the starting index in the enabled arrays.
  - count - Specifies the number of indices to be rendered.



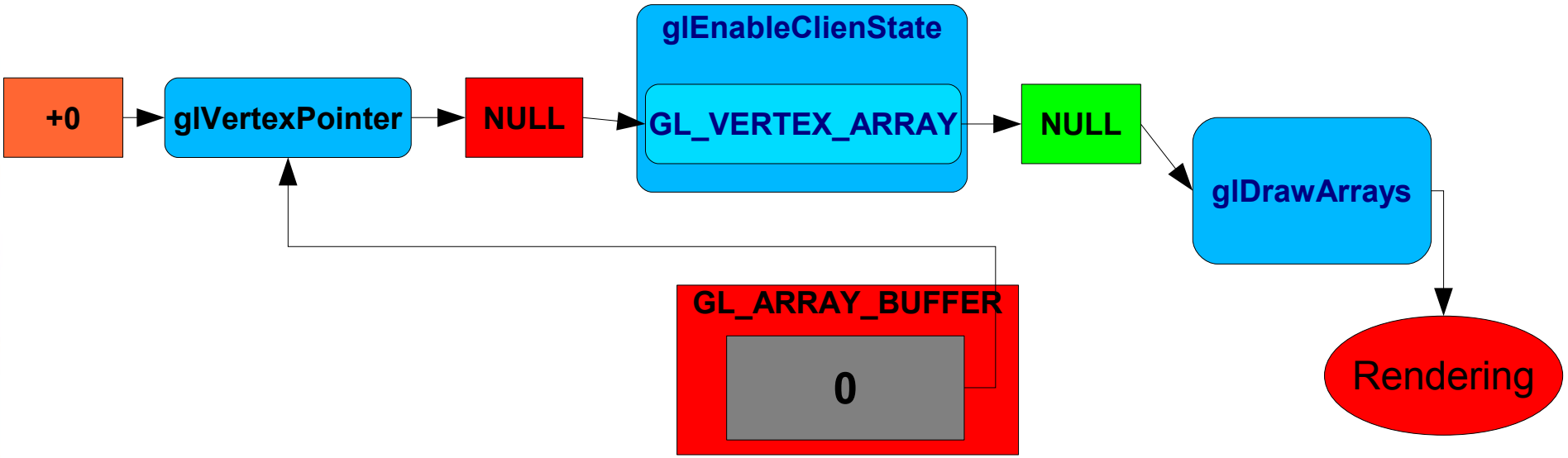
Rendering ARRAY_BUFFER
(VBO)

- void **glVertexPointer**(GLint size, GLenum type,
  GLsizei stride, const GLvoid *pointer)
  - pointer == NULL (0) ???



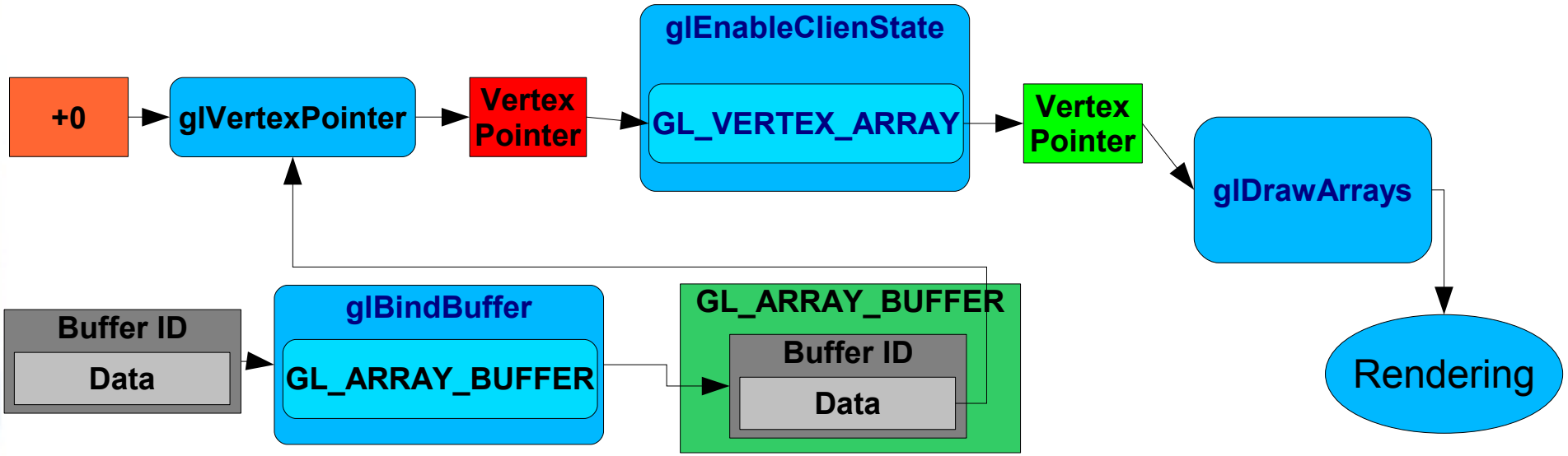
Rendering ARRAY_BUFFER
(VBO)

- **glVertexPointer** is retrieving pointer from GL_ARRAY_BUFFER



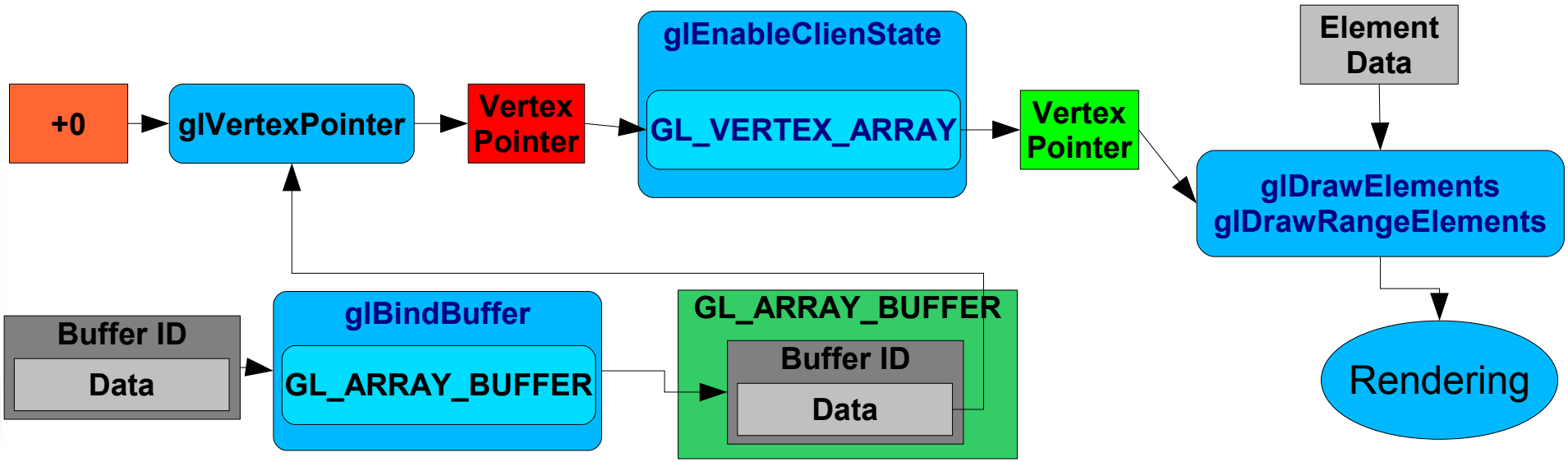
Rendering ARRAY_BUFFER
(VBO)

- Bind valid glBufferID, generated by **glGenBuffers** to GL_ARRAY_BUFFER
  - Similar for
    - **glVertexNormal**, **glVertexColor**,...



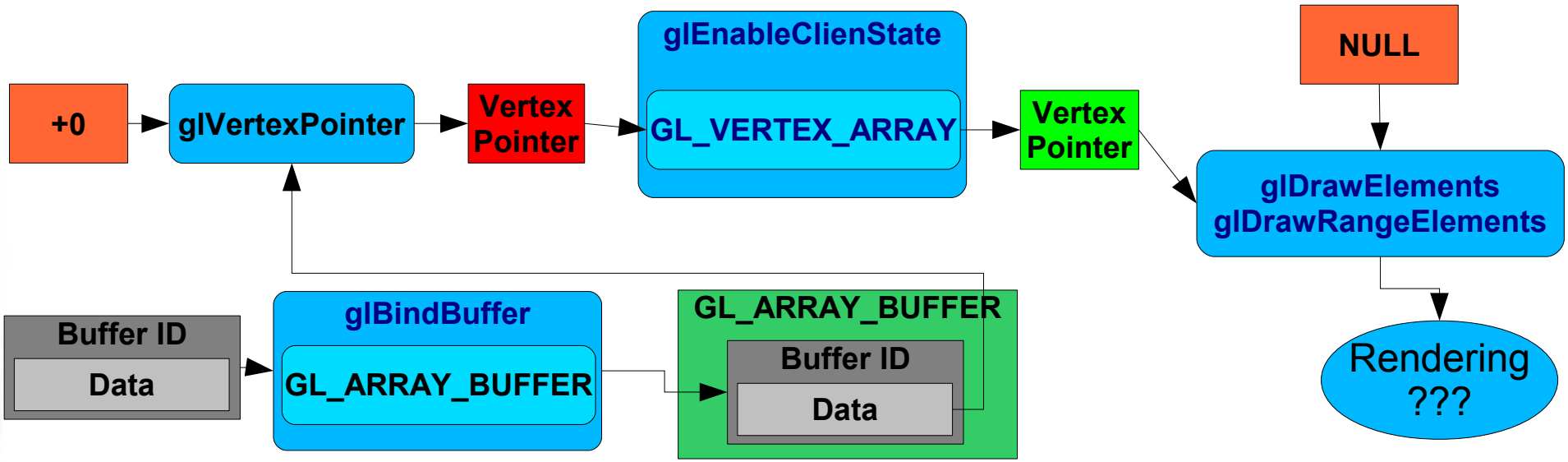
Rendering ARRAY_BUFFER
with glDrawElements

- Bind valid glBufferID, generated by **glGenBuffers** to GL_ARRAY_BUFFER
  - Similar for
    - **glVertexNormal**, **glVertexColor**,...



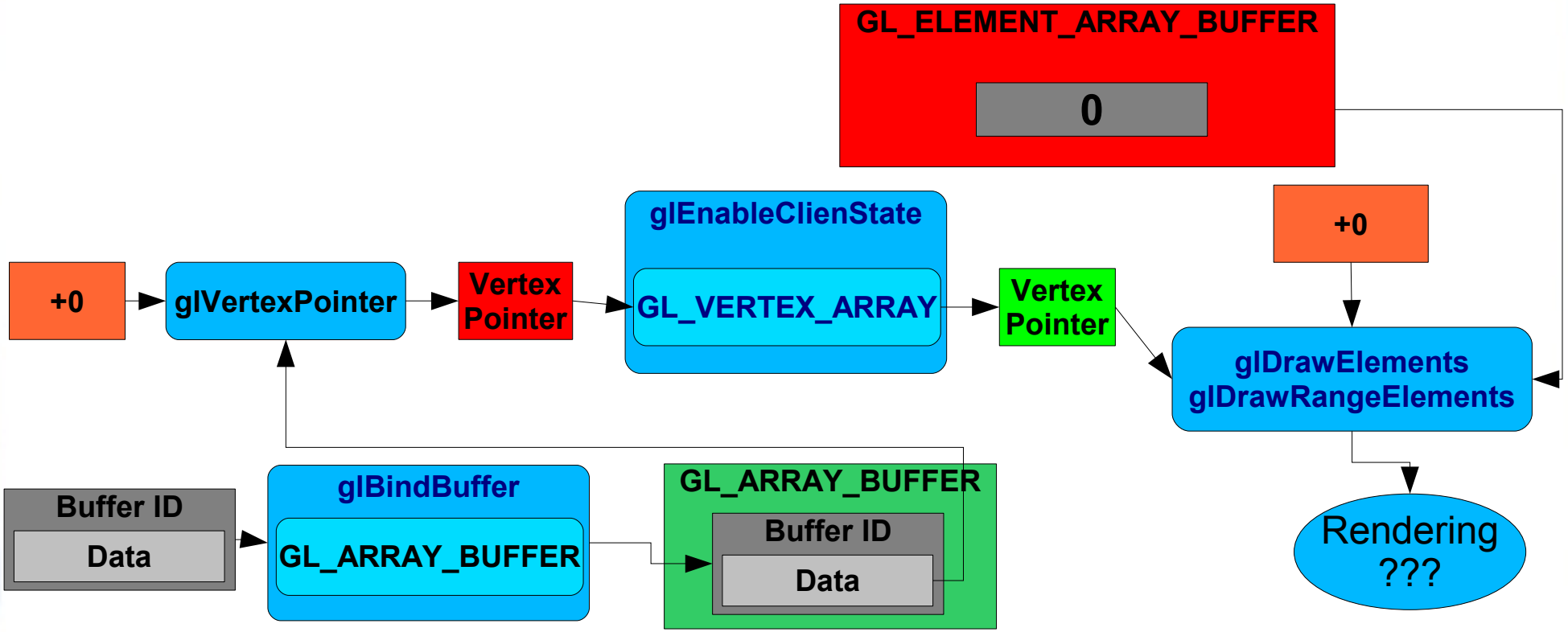
Rendering ELEMENT_ARRAY_BUFFER

- void **glDrawElements**(GLenum mode,
  - GLsizei count,
  - GLenum type,
  - const GLvoid *indices)
  - Indices == NULL (0) ???



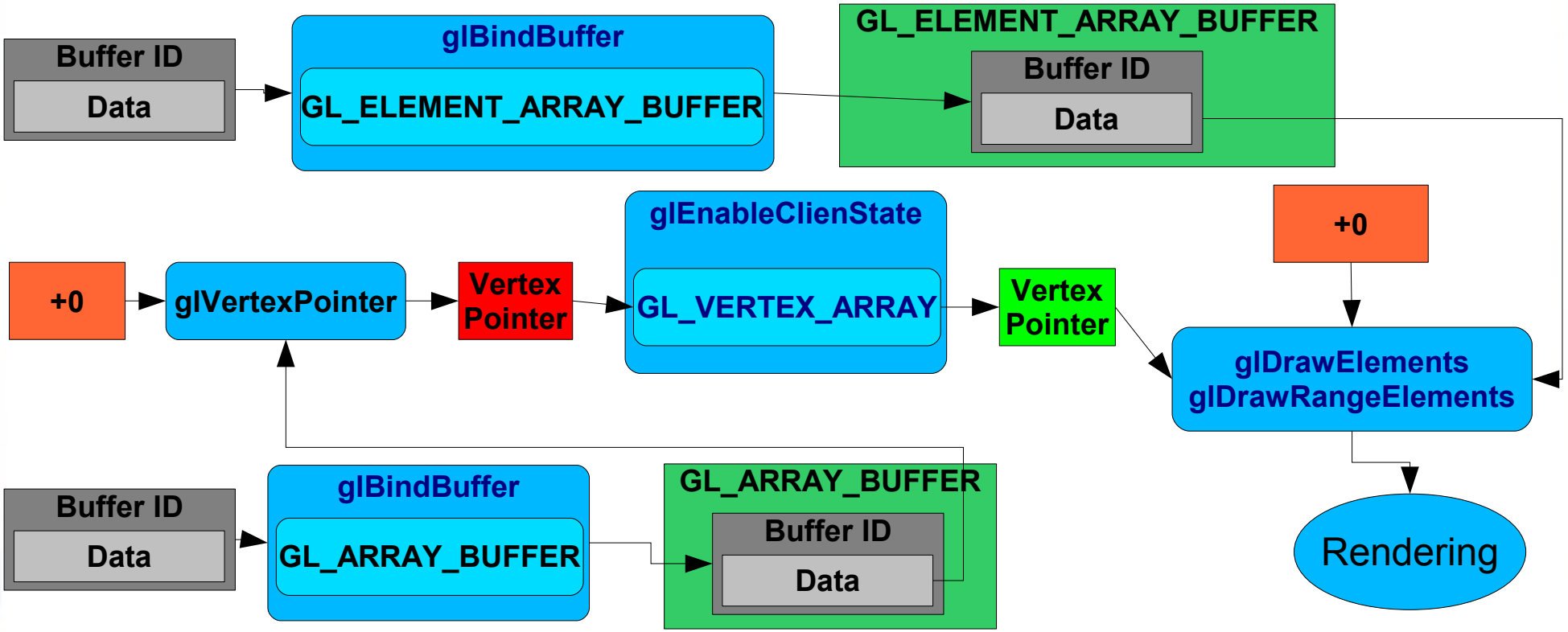
Rendering ELEMENT_ARRAY_BUFFER

- **glDrawElements** is retriveing indices from
  - GL_ELEMENT_ARRAY_BUFFER



Rendering ELEMENT_ARRAY_BUFFER

- Bind valid glBufferID, generated by **glGenBuffers** to GL_ELEMENT_ARRAY_BUFFER



FBO

- void glGenFramebuffersEXT(GLsizei n, GLuint * framebuffers);
  - generate framebuffer object names
- void glDeleteFramebuffersEXT(GLsizei n, const GLuint * framebuffers);
  - delete named framebuffer objects
- void glBindFramebufferEXT(GLenum target, GLuint framebuffer);
  - bind a named framebuffer object
  - target must be GL_FRAMEBUFFER_EXT

Texture Attachments
(rendering to texture)

- void glFramebufferTexture2DEXT(GLenum target, GLenum attachment, GLenum textarget, GLuint texture, Glint level);
  - target – must be GL_FRAMEBUFFER_EXT
  - attachment –
    - GL_COLOR_ATTACHMENT0 .. 3 – four color textures
    - GL_DEPTH_ATTACHMENT – one depth texture
    - GL_STENCIL_ATTACHMENT – one stencil texture
  - textarget –
    - GL_TEXTURE_2D
    - GL_TEXTURE_CUBE_MAP_POSITIVE_X, _Y, _Z,
    - GL_TEXTURE_CUBE_MAP_ NEGATIVE_X, _Y, _Z
  - texture – texture name generated by glGenTextures
  - level – mipmap level to attach from texture to attachment

Renderbuffers
(render attachments for FBOs)

- void glGenRenderbuffersEXT (GLsizei n, GLuint * renderbuffs)
  - generate renderbuffer object names
- void glDeleteRenderbuffersEXT (GLsizei n, const GLuint * renderbuffs);
  - delete named renderbuffer objects
- void glBindRenderbufferEXT (GLenum target, GLuint renderbuffer)
  - bind a named renderbuffer object
- void glRenderbufferStorageEXT (GLenum target, GLenum internalformat, GLsizei width, GLsizei height)
  - target – must be GL_RENDERBUFFER_EXT
  - internalformat - GL_RGBA4, GL_RGB565, GL_RGB5_A1, GL_DEPTH_COMPONENT16, 24, GL_STENCIL_INDEX8,...

Renderbuffer attachments

- void glFramebufferRenderbufferEXT (GLenum target, GLenum attachment, GLenum renderbuffertarget, GLuint renderbuffer);
  - target – must be GL_FRAMEBUFFER_EXT
  - attachment -
    - GL_COLOR_ATTACHMENT0..3
    - GL_DEPTH_ATTACHMENT
    - GL_STENCIL_ATTACHMENT
  - renderbuffertarget – must be GL_RENDERBUFFER_EXT
  - renderbuffer - name generated by glGenRenderbuffersEXT

Creating FBO and RenderBuffers

- // create Detph render buffer for depth buffer (depth-testing) when using FBO
- glGenRenderbuffersEXT(1, &gFBdepthRB);
- glBindRenderbufferEXT(GL_RENDERBUFFER_EXT, gFBdepthRB);
- glRenderbufferStorageEXT(GL_RENDERBUFFER_EXT, GL_DEPTH_COMPONENT24, FB_WIDTH,FB_HEIGHT);
- // create FrameBuffes
- glGenFramebuffersEXT(1, &gFB);

Attach textures and renderbuffers

- glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, gFB);
- // attach Detph buffer
- glFramebufferRenderbufferEXT(GL_FRAMEBUFFER_EXT,
  GL_DEPTH_ATTACHMENT_EXT,
  GL_RENDERBUFFER_EXT, gFBdepthRB);
- // attach Depth texture to retrive depth (render) buffer for future processing
- glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
  GL_DEPTH_ATTACHMENT_EXT, GL_TEXTURE_2D,
  gFBDepthTexID, 0);
- // attach Color texture to retrive color buffer for future processing
- glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
  GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D,
  gFBColorTexID, 0);

Rendering with FBO

- glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, gFB);
- // draw to color_attachment0 of the bound FBO
- glDrawBuffer(GL_COLOR_ATTACHMENT0_EXT);
  - // draw to multiple color attachments of the bound FBO
  - Gluint buffers[ ]={ GL_COLOR_ATTACHMENT0_EXT, GL_COLOR_ATTACHMENT1_EXT};
  - glDrawBuffers(2,buffers);
- … your desired rendering method …
- // unbind FBO – rendering to standard frame buffer (window)
- glBindFramebufferEXT (GL_FRAMEBUFFER_EXT, 0);

Create textures for FBO

- // color texture
- glGenTextures(1,&gFBColorTexID);
- glBindTexture(GL_TEXTURE_2D,gFBColorTexID);
- // Linear Filtering
- glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
- glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
- // storage type
- glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, FB_WIDTH, FB_HEIGHT, 0, GL_RGBA, GL_UNSIGNED_BYTE, NULL);
- // Depth texture to get also the depth buffer into texture
- … same as color texture
- // storage type
- glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, FB_WIDTH, FB_HEIGHT,0,GL_DEPTH_COMPONENT, GL_UNSIGNED_BYTE, NULL);

References

- OpenGL® Extension Registry
  - http://opengl.org/registry/
- OpenGL ES 2.0 Reference Pages
  - http://www.khronos.org/opengles/sdk/docs/man/