Contents

1 Core Functions and Basic Configuration

1.1 Core Functions

- `glEnable`, `glDisable`, `glIsEnabled`
  Controls server-side features.

- `glEnableClientState`, `glDisableClientState`
  Controls client-side features.

- `glGetBoolean`, `glGetInteger`, `glGetFloat`, `glGetDouble`, `glGetString`
  Reads server-side state variables.

- `glFlush`
  Flushes commands to the server.

- `glFinish`
  Flushes commands to the server and waits for them to finish.

- `glGetError`
  Returns the last error.

- `glPushAttrib`, `glPopAttrib`
  Push/pop server-side state.

- `glPushClientAttrib`, `glPopClientAttrib`
  Push/pop client-side state.

1.2 Viewport Configuration

- `glViewport`
  Defines the transformation from normalized device coordinates to window coordinates.

- `glViewportIndexed`
  Like `glViewport`, but takes the index of the viewport to configure.

- `glViewportArray`
  Like `glViewport`, but configures multiple viewports at once.

- `glScissor`
  Defines a *scissor box* in the viewport (using window coordinates) and restricts drawing to that box.

- `glScissorIndexed`
  Like `glScissor`, but takes the index of the viewport to configure.

- `glScissorArray`
  Like `glScissor`, but configures multiple viewports at once.

- `glDepthRange`
  Defines the mapping of depth values to internal depth values for the depth buffer.

- `glDepthRangeIndexed`
  Like `glDepthRange`, but takes the index of the viewport to configure.

- `glDepthRangeArray`
  Like `glDepthRange`, but configures multiple viewports at once.

1.3 Misc. Configuration

- `glCullFace`
  Switches between front-face culling and back-face culling.

- `glFrontFace`
  Specifies which polygon side is the "front" face.

- `glHint`
  Sets rendering quality hints.

2 Render State Configuration

2.1 Transform Management

(`glVertex`) → Object Coordinates
→ (ModelView Matrix) → Eye Coordinates
→ (Projection Matrix) → Clip Coordinates
→ (Divide by w) → Normalized Device Coordinates (+-1 cube)
→ (Viewport Transform) → Window Coordinates (0,0..w,h)

2.1.1 General

- `glMatrixMode`
  Selects whether the modelview matrix, projection matrix or texture matrix is affected by future calls.

- `glPushMatrix`
  Pushes the current matrix onto the matrix stack for the current mode (each mode has a separate stack).

- `glPopMatrix`
  Pops the top of the matrix stack for the current mode.

2.1.2 Loading Matrices

- `glLoadIdentity`
  Loads the identity matrix.

- `glLoadMatrix`
  Loads the specified matrix.

- `glLoadTransposeMatrix`
  Loads the transpose of the specified matrix.

2.1.3 Multiplying Matrices

- `glTranslate{f,d}`
  Multiplies the current matrix with a translation matrix.

- `glRotate{f,d}`
  Multiplies the current matrix with a rotation matrix.

- `glScale{f,d}`
  Multiplies the current matrix with a scaling matrix.

- `glFrustum`
  Multiplies the current matrix with a perspective projection matrix.

- `glOrtho`
  Multiplies the current matrix with a parallel projection matrix.

- `glMultMatrix`
  Multiplies the current matrix with the specified matrix.

- `glMultTransposeMatrix`
  Multiplies the current matrix with the transpose of the specified matrix.

2.2 Rasterizing

2.2.1 Color Buffer and Blending

- `glAlphaFunc`
  Discards pixels based on comparing the alpha value with a fixed reference value (performance!)

- `glBlendFunc`, `glBlendEquation`,
  Defines how pixels are blended into the color buffer.

- `glBlendFuncSeparate`, `glBlendEquationSeparate`
  "set the RGB blend equation and the alpha blend equation separately"

- `glBlendColor`
  Sets the auxiliary color for `glBlendFunc`.

- `glColorMask`, `glColorMaski`
  Blocks off writing to specific color channels.

2.2.2 Depth Buffer

- `glDepthFunc`
  Discards pixels based on comparing the depth value with the depth value from the frame buffer (Z-/W-buffering)

- `glDepthMask`
  Enables / disables writing to the depth buffer.

2.2.3 Stencil Buffer

- `glStencilFunc`
  Discards pixels based on comparing a fixed stencil value with the value from the stencil buffer, masking both values with a defined bit mask.

- `glStencilMask`
  Defines the write-enable bit mask when writing to the stencil buffer.

- `glStencilOp`
  Defines the effects of (stencil-rejected, stencil-accepted-depth-rejected and both-accepted) pixels on the stencil buffer.

- `glStencilFuncSeparate`, `glStencilMaskSeparate`, `glStencilOpSeparate`
  Configures stencil buffer behavior separately for front-facing and back-facing polygons.

2.2.4 Misc. Rasterizing Configuration

- `glSampleCoverage`
  Specifies a bit mask when multisampling each pixel.

2.3 Effects

2.3.1 Lighting

- `glLight`
  "set light source parameters"

- `glGetLight`
  "return light source parameter values"

- `glLightModel`
  "set the lighting model parameters"

- `glMaterial`
  "specify material parameters for the lighting model"

- `glGetMaterial`
  "return material parameters values"
- `glColorMaterial`
  "cause a material color to track the current color"
- `glShadeModel`
  Switches between flat shading and smooth shading.
- `glMinSampleShading`
  "specifies minimum rate at which sample shading takes place"

2.3.2 Fog

- `glFog`
- `glFogCoord`
- `glFogCoordPointer`

2.3.3 Custom Clipping Planes

- `glClipPlane`
  Controls custom clipping planes.
- `glGetClipPlane`
  Controls custom clipping planes.

2.3.4 Decals

- `glPolygonOffset`
  "set the scale and units used to calculate depth values" (used for outlining and decals)

2.3.5 Points and Lines

- `glPolygonMode`
  Switches between polygon drawing, outline drawing, and vertex drawing.
- `glPolygonStipple` ...

- `glGetPolygonStipple` ...
- `glLineWidth`
  Specifies the line width for line / outline drawing.
- `glLineStipple` ...
- `glPointSize`
  Specifies the point size for point / vertex drawing.
- `glPointParameter`
  Specifies parameters for point / vertex drawing.

3 Primitives, Arrays and Buffers

3.1 Direct Mode Rendering

- `glBegin(type)`, `glEnd`
  Start / stop drawing primitives of type `type`.

- `glVertex{2,3,4}{i,f,d}`
  Send a vertex for the current primitive. Default values are (x, y, 0, 1)

- `glNormal3{b,i,f,d}`
  Set the normal for the next vertex.

- `glColor{3,4}{ub,b,f,d}`, `glColorP{3,4}{u,ui}`
  Set the color for the next vertex.

- `glSecondaryColor*`
  Sets a secondary per-vertex color that gets added to each pixel.

- `glTexCoord{1,2,3,4}{f,d}`
  Set the texture coordinates for the next vertex.

- `glMultiTexCoord`
  Like `glTexCoord`, but takes the target texture unit as an additional argument.

- `glEdgeFlag`
  Hides specific edges when outlining polygons because they're "inner" tesselation edges.

- `glRect`
  Draws a rectangle. Equivalent to sending four vertices, surrounded by `glBegin` / `glEnd`.

3.2 Buffers

3.2.1 Life Cycle

- `glGenBuffers`
  Generates one or more buffer names.

- `glDeleteBuffers`
  Deletes one or more buffer names and associated buffers.

- `glIsBuffer`
  Check if a buffer name is associated with a buffer.

3.2.2 Binding Buffers

- `glBindBuffer`
  Binds a buffer to a target for future calls and for drawing, creating the buffer if necessary.

- `glBindBufferBase`
  bind a buffer object to an indexed buffer target (target with multiple bind points)

- `glBindBufferRange`
  bind a range within a buffer object to an indexed buffer target (target with multiple bind points)

3.2.3 Handling Buffer Data

- `glClearBufferData`, `glClearBufferSubData`
  Fills (part of) a buffer with a fixed value.

- `glBufferData`
  Defines the usage mode for a buffer and uploads buffer data.

- `glBufferSubData`
  Updates part of a buffer.

- `glMapBuffer`, `glMapBufferRange`, `glUnmapBuffer`, `glGetBufferPointer`, `glFlushMappedBufferRange`
  Maps / unmaps (part of) a buffer to the client's address space; obtains a pointer to the mapped data; flushes changes to a mapped buffer.

- `glCopyBufferSubData`
  "copy part of the data store of a buffer object to the data store of another buffer object"

- `glInvalidateBufferData`, `glInvalidateBufferSubData`
  Invalidates (part of) a buffer.

- `glGetBufferSubData`
  Reads part of a buffer.

- `glGetBufferParameter`
  Reads a buffer parameter.

3.3 Using Arrays and Buffers

(Client-side) arrays and (server-side) buffers are configured using the same functions. For each pointer, if a corresponding buffer is bound, then it is used (and the "pointer" is an offset for that buffer), otherwise a client-side array is used. Both arrays and buffers are affected by `glEnableClientState` and `glDisableClientState` regarding arrays.

3.3.1 Configuration

- `glVertexPointer`
  Selects a vertex array or sets the offset for a vertex buffer.

- `glNormalPointer`
  Selects a normal array or sets the offset for a normal buffer.

- `glColorPointer`
  Selects a per-vertex color array or sets the offset for a per-vertex color buffer.

- `glSecondaryColorPointer`
  Selects a per-vertex secondary color array or sets the offset for a per-vertex secondary color buffer.

- `glTexCoordPointer`
  Selects a texture coordinate array or sets the offset for a texture coordinate buffer.

- `glEdgeFlagPointer`
  Selects a edge flag array or sets the offset for a edge flag buffer.

- `glInterleavedArrays`
  Deprecated function to set multiple client-side pointers at once.

- `glGetPointer`
  Returns a client-side array pointer or server-side buffer offset.

3.3.2 Drawing Primitives From Arrays and Buffers

- `glDrawArrays`
  Draws primitives from the previously defined arrays.

- `glArrayElement`
  Draws the vertex from index i of the current array / buffer (requires `glBegin` / `glEnd`).

- `glDrawElements`
  Draws primitives from the previously defined arrays as well as an additional index array that associates primitives with (re-used) vertices.

- `glDrawElementsBaseVertex`
  Like `glDrawElements`, with a constant added on-the-fly to each element of the index array.

- `glDrawRangeElements`
  Like `glDrawElements` but with additional lower and upper bound hints for the indices from the index array.

- `glDrawRangeElementsBaseVertex`
  Like `glDrawRangeElements`, with a constant added on-the-fly to each element of the index array.

- `glMultiDrawArrays`, `glMultiDrawElements`
  Performs multiple `glDrawArrays` / `glDrawElements` invocations at once, taking the parameters for each invocation from an array.

- `glPrimitiveRestartIndex` Sets the "primitive restart index", that is, an array index that doesn't issue a vertex but instead starts a new draw operation.

3.3.3 Instancing Support for Shaders

These functions repeat the above functions n times, incrementing a counter each time. They are useless without a geometry shader that interprets the counter, since they otherwise just repeatedly draw the same object. A shader could, for example, apply a different transformation each time to draw multiple instances at different locations.

- `glDrawArraysInstanced`
  Repeats `glDrawArrays` n times.

- `glDrawElementsInstanced`
  Repeats `glDrawElements` n times.

- `glDrawArraysInstancedBaseInstance`
  Like `glDrawArraysInstanced`, but respects "instanced" generic vertex attributes.

- `glDrawElementsInstancedBaseInstance`
  Like `glDrawElementsInstanced`, but respects "instanced" generic vertex attributes.

- `glDrawElementsInstancedBaseVertex`
  Like `glDrawElementsInstanced`, with a constant added on-the-fly to each element of the index array.

- `glDrawElementsInstancedBaseVertexBaseInstance`
  Like `glDrawElementsInstanced`, with a constant added on-the-fly to each element of the index array AND respects "instanced" generic vertex attributes.

- `glDrawArraysIndirect`
  Variant of `glDrawArraysInstancedBaseInstance` that takes a pointer to the remaining function arguments.

- `glDrawElementsIndirect`
  Variant of `glDrawElementsInstancedBaseVertexBaseInstance` that takes a pointer to the remaining function arguments.

- `glMultiDrawArraysIndirect`, `glMultiDrawElementsIndirect`
  Performs multiple `glDrawArraysInstancedBaseInstance` / `glDrawElementsInstancedB`

invocations at once, taking the parameters for each invocation from an array.

4 Textures, Images and Bitmaps

4.1 Dealing With Image Data

4.1.1 Pixel Handling Configuration

Affects texture uploading, texture downloading, drawing pixels and bitmaps, etc.

- `glPixelStore{i,f}` Defines how pixels are stored in application memory.
- `glPixelTransfer{i,f}` Defines a per-channel scale and offset when reading, writing or copying pixels.
- `glPixelMap{usv,uiv,fv}` Defines a per-channel mapping, e.g. for palette lookup or gamma correction.
- `glGetPixelMap` Reads the pixel map.

4.2 Handling Textures

4.2.1 Life Cycle

- `glGenTextures`
  Generates one or more texture names.
- `glDeleteTextures`
  Deletes texture names and associated textures.
- `glIsTexture`
  Checks whether a texture name is associated with a texture.

4.2.2 Binding Textures

- `glBindTexture`
  Binds a texture for future calls and for drawing. The target for the first binding of a texture determines its dimension. Separate bindings exist per target.

4.2.3 Configuring Textures

- `glTexStorage*`, `glTexStorage*Multisample`
  Specifies the storage for all levels of a texture at once.

- `glGenerateMipmap`
  "generate mipmaps for a specified texture target"

4.2.4 Handling Texture Data

- `glTexImage{1D,2D,3D}`, `glTexImage{2D,3D}Multisample`
  Upload texture image data.

- `glTexSubImage{1D,2D,3D}`
  Update part of a texture image.

- `glCompressedTexImage{1D,2D,3D}`
  Upload texture image data from a compressed source.

- `glCompressedTexSubImage{1D,2D,3D}`
  Update part of a texture image from a compressed source.

- `glInvalidateTexImage`, `glInvalidateTexSubImage`
  Invalidate (part of) a texture.

- `glGetTexImage`
  Download texture image data.

- `glGetCompressedTexImage`
  "return a compressed texture image"

- `glTextureView`
  "initialize a texture as a data alias of another texture's data store"

- `glCopyImageSubData`
  "perform a raw data copy between two images"

4.2.5 Texture-Related Render State

- `glActiveTexture`, `glClientActiveTexture`
  Selects the active texture unit. The server-side active unit affects future server-side operations and drawing; the client-side active unit affects future client-side operations (e.g. `glTexCoordPointer`).
- `glTexParameter(target, parameterName, parameterValue`
  Set texture parameters.
- `glGetTexParameter`
  Get texture parameters.
- `glGetTexLevelParameter(target, level, parameterName, parameterValue`
  Get LOD-(mipmap-)dependent texture parameters.
- `glTexEnv`
  ???
- `glGetTexEnvf`
  ???
- `glTexGen`
  Defines automatic generation of texture coordinates.
- `glGetTexGen`
  Returns automatic texture generation parameters.
- `glAreTexturesResident`
  "determine if textures are loaded in texture memory"

4.3 Samplers

4.3.1 Life Cycle

- `glGenSamplers`
  "generate sampler object names"
- `glDeleteSamplers`
  "delete named sampler objects"
- `glIsSampler`
  "determine if a name corresponds to a sampler object"

4.3.2 Binding Samplers

- `glBindSampler`
  "bind a named sampler to a texturing target"

4.3.3 Configuring Samplers

- `glSamplerParameter`
  Specifies sampler parameters.

- `glGetSamplerParameter`
  Returns sampler parameters.

4.4 Connecting Textures and Buffers

- `glTexBuffer`
  "attach the storage for a buffer object to the active buffer texture"

- `glTexBufferRange`
  "bind a range of a buffer's data store to a buffer texture"

4.5 Drawing Bitmaps and Pixels

- `glRasterPos{2,3,4}{s,i,f,d}`
  Sets the raster position by transforming 3d coordinates like for `glVertex`.

- `glWindowPos`
  Sets the raster position directly using window coordinates.

- `glBitmap`
  Draws a bitmap, with a "1" bit using the current color and a "0" bit being transparent, to the current raster position. This function also updates the raster position.

- `glDrawPixels`
  Copies pixels from memory to the current raster position of the framebuffer.

- `glCopyPixels`
  Copies pixels from the specified framebuffer position to the current raster position.

- `glPixelZoom` Specifies the zoom factor for `glDrawPixels` and `glCopyPixels`.

5 OpenGL (non-shader part)

5.1 Display Lists

- `glGenLists`
  Generates display lists.
- `glDeleteLists`
  Deletes display lists.
- `glIsList`
  "determine if a name corresponds to a display list"
- `glNewList`
  Start filling a display list.
- `glEndList`
  Finish filling a display list.
- `glCallList`
  Executes a display list.
- `glCallLists`
  Executes multiple display lists.
- `glListBase`
  Defines an offset for display list indices when calling lists.

5.2 Splines, B-Splines, NURBS, Bezier Curves, etc.

- `glMapGrid{1,2}{f,d}`
- `glMap{1,2}{f,d}`
- `glGetMap{1,2}{f,d}`
- `glEvalPoint{1,2}`
- `glEvalCoord{1,2}{f,d,fv,dv}`
- `glEvalMesh{1,2}`

5.3 Synchronization

Used to synchronize client and server processes.

- `glClientWaitSync`
- `glFenceSync`
- `glIsSync`
- `glWaitSync`
- `glGetSync`
- `glDeleteSync`
- `glObjectPtrLabel`
- `glGetObjectPtrLabel`
- `glMemoryBarrier`

5.4 Frame Buffer Management

- `glClear`
  Clears one or more buffers.
- `glClearColor`
  Set the value that `glClear` will use for the color buffer.
- `glClearDepth`
  Set the value that `glClear` will use for the depth buffer.
- `glClearStencil`
  Set the value that `glClear` will use for the stencil buffer.
- `glClearAccum`
  Set the value that `glClear` will use for the accumulation buffer.
- `glAccum`
  Transfer between color buffer and accumulation buffer.
- `glDrawBuffer`
  Selects the buffer (double buffering, stereo rendering)

- `glInvalidateFramebuffer`, `glInvalidateSubFramebuffer`
  Invalidates (part of) the frame buffer.
- `glGenFramebuffers`
  ...
- `glDeleteFramebuffers`
  ...
- `glBindFramebuffer`
  ...
- `glIsFramebuffer`
  ...
- `glCheckFramebufferStatus`
  ...
- `glFramebufferTexture`
  "attach a level of a texture object as a logical buffer to the currently bound framebuffer object"
- `glFramebufferTextureLayer`
  "attach a single layer of a texture to a framebuffer"
- `glClearBuffer`
  "clear individual buffers of the currently bound draw framebuffer"
- `glFramebufferParameter`
  "set a named parameter of a framebuffer"
- `glGetFramebufferAttachmentParameter`
  "retrieve information about attachments of a bound framebuffer object"
- `glFramebufferParameter`
  "set a named parameter of a framebuffer"
- `glGetFramebufferParameter`
  "retrieve a named parameter from a framebuffer"
- `glGetMultisample`
  "retrieve the location of a sample"

5.5 Renderbuffers

- `glGenRenderbuffers`
- `glDeleteRenderbuffers`
- `glIsRenderbuffer`
- `glBindRenderbuffer`
- `glGetRenderbufferParameter`
- `glRenderbufferStorage`
- `glRenderbufferStorageMultisample`
- `glFramebufferRenderbuffer`
  "attach a renderbuffer as a logical buffer to the currently bound framebuffer object"
- `glSampleMaski`
  "set the value of a sub-word of the sample mask"

5.6 Reading the Frame Buffer

- `glReadBuffer`
  Selects the frame buffer to read from.
- `glCopyTexImage{1,2}D`
  Defines a texture by copying a rectangular part of the frame buffer.
- `glCopyTexSubImage{1,2,3}D`
  Updates a texture by copying a rectangular part of the frame buffer.

- `glReadPixels`
  Copies pixels from the framebuffer into memory.
- `glClampColor`
  "specify whether data read via glReadPixels should be clamped"
- `glBlitFramebuffer`
  Copies pixels within the frame buffer or from one frame buffer to another.

5.7 Render Queries

- `glGenQueries`
  "generate query object names"
- `glDeleteQueries`
  "delete named query objects"
- `glIsQuery`
  "determine if a name corresponds to a query object"
- `glGetQueryObject`
  "return parameters of a query object"
- `glGetQueryIndexed`
  "return parameters of an indexed query object target"
- `glBeginQuery`, `glEndQuery`
  "delimit the boundaries of a query object"
- `glBeginQueryIndexed`, `glEndQueryIndexed`
  "delimit the boundaries of a query object on an indexed target"
- `glGetQuery`
  "return parameters of a query object target"
- `glQueryCounter`
  "record the GL time into a query object after all previous commands have reached the GL server but have not yet necessarily executed"
- `glBeginConditionalRender`, `glEndConditionalRender`
  Execute GL commands only if a query detects that samples passed the tests.

5.8 Debugging

- `glDebugMessageControl`
  "control the reporting of debug messages in a debug context"
- `glPushDebugGroup`
  "push a named debug group into the command stream"
- `glPopDebugGroup`
  "pop the active debug group"
- `glDebugMessageCallback`
  "specify a callback to receive debugging messages from the GL"
- `glDebugMessageInsert`
  "inject an application-supplied message into the debug message queue"
- `glGetDebugMessageLog`
  "retrieve messages from the debug message log"
- `glObjectLabel`
  "label a named object identified within a namespace"
- `glGetObjectLabel`
  "retrieve the label of a named object identified within a namespace"

5.9 Feedback Buffers (structured rendering log, selection)

- `glInitNames`
  ...
- `glLoadName`
  ...
- `glPushName`
  ...
- `glPopName`
  ...

- `glFeedbackBuffer`
  ...
- `glSelectBuffer`
  ...
- `glRenderMode`
  ...
- `glPassThrough`
  ...

5.10 Various Rendering Parameters

- `glPrioritizeTextures`
  Sets texture priority with respect to eviction from the graphics memory.
- `glGetInternalformat`
  "retrieve information about implementation-dependent support for internal formats"
- `glPatchParameter`
  "specifies the parameters for patch primitives" (hardware tesselation control)

6 OpenGL Shaders

- glCreateShader
- glShaderSource
- glCompileShader
- glCreateProgram
- glAttachShader
- glLinkProgram
- glUseProgram
- glProgramUniform, glProgramUniformMatrix
- glGetProgramiv
- glValidateProgram
- glValidateProgramPipeline
- glGetProgramBinary
- glGetProgramResourceLocation
- glGetProgramStage
- glGetProgram
- glCreateShaderProgram
- glGetProgramPipeline.
- glGetProgramPipelineInfoLog
- glGetProgramInterface.
- glIsProgram
- glDeleteProgramPipelines
- glDeleteProgram

- glGetProgramResourceIndex
- glGetProgramResourceLocationIndex
- glActiveShaderProgram
- glGetProgramStagei
- glBindProgramPipeline
- glGetProgramResource
- glGetProgramResourceName
- glGenProgramPipelines
- glGetProgramInfoLog
- glProgramParameteri
- glUseProgramStages
- glProgramBinary
- glIsProgramPipeline
- glGetShaderInfoLog
- glDeleteShader
- glDetachShader
- glShaderBinary
- glGetAttachedShaders
- glGetShader
- glGetShaderSource
- glGetShaderPrecisionFormat
- glShaderStorageBlockBinding
- glIsShader

- glGetShaderi
- glReleaseShaderCompiler
- glBindAttribLocation
- glGetUniform
- glGetUniformLocation
- glGetActiveAttrib
- glGetActiveAttribSize
- glGetActiveAttribType
- glGetActiveUniform
- glGetAttribLocation
- glVertexAttrib
- glGetVertexAttrib
- glVertexAttribPointer
- glGetVertexAttribPointer
- glUniform
- glUniformMatrix
- glUniformSubroutinesu
- glGetUniformBlockIndex
- glGetActiveSubroutineUniform
- glGetActiveSubroutineUniformName
- glGetActiveUniformType
- glGetActiveUniformSize
- glGetActiveUniformBlockName

- glGetSubroutineUniformLocation
- glGetUniformIndices
- glGetActiveUniforms
- glUniformBlockBinding
- glGetActiveUniformBlock
- glGetActiveUniformName
- glGetSubroutineIndex
- glGetActiveSubroutineName
- glVertexAttribFormat
- glVertexAttribDivisor
- glVertexAttribBinding
- glEnableVertexAttribArray
- glDisableVertexAttribArray
- glDrawBuffers
- glVertexBindingDivisor
- glBindFragDataLocation
- glBindFragDataLocationIndexed
- glGetActiveAtomicCounterBuffer
- glBindImageTexture
- glGetUniformSubroutine
- glDispatchCompute
- glDispatchComputeIndirect
- glProvokingVertex

- `glGetFragDataIndex`
- `glGetFragDataLocation`
- `glBindVertexBuffer`
  "bind a buffer to a vertex buffer bind point" (only used for generic vertex attributes)

6.1 Vertex Array Objects (related to generic vertex attributes)

- `glGenVertexArrays`
  "generate vertex array object names"
- `glDeleteVertexArrays`
  "delete vertex array objects"
- `glIsVertexArray`
  "determine if a name corresponds to a vertex array object"
- `glBindVertexArray`
  "bind a vertex array object"

6.2 Transform Feedback

- `glGenTransformFeedbacks`
- `glDeleteTransformFeedbacks`
- `glBeginTransformFeedback`
- `glEndTransformFeedback`
- `glBindTransformFeedback`
- `glDrawTransformFeedback`
- `glDrawTransformFeedbackInstanced`
- `glDrawTransformFeedbackStream`
- `glDrawTransformFeedbackStreamInstanced`

- glGetTransformFeedbackVarying
- glIsTransformFeedback
- glPauseTransformFeedback
- glResumeTransformFeedback
- glTransformFeedbackVaryings

7 Useless OpenGL Features

7.1 Functions Related to Indexed Color Modes

- `glIndexPointer`
- `glClearIndex`
- `glLogicOp`