

GLAPI/glVertexAttribPointer

From OpenGL.org

glVertexAttribPointer: define an array of generic vertex attribute data. Examples are available.

glVertexAttribPointer

Core in version	4.3
Adopted into core in version	2.0

Contents

- 1 Function Definition
- 2 Description
- 3 Notes
- 4 Errors
- 5 Associated Gets
- 6 See Also
- 7 Copyright

Function Definition

```
void glVertexAttribPointer(GLuint index, GLint size, GLenum type, GLboolean normalized, GLsizei stride, const GLvoid * pointer);
void glVertexAttribIPointer(GLuint index, GLint size, GLenum type, GLsizei stride, const GLvoid * pointer);
void glVertexAttribLPointer(GLuint index, GLint size, GLenum type, GLsizei stride, const GLvoid * pointer);
```

index

Specifies the index of the generic vertex attribute to be modified.

size

Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, 4. Additionally, the symbolic constant `GL_BGRA` is accepted by **glVertexAttribPointer**. The initial value is 4.

type

Specifies the data type of each component in the array. The different functions take different values.

glVertexAttribPointer and **glVertexAttribIPointer** both take: `GL_BYTE`, `GL_UNSIGNED_BYTE`, `GL_SHORT`, `GL_UNSIGNED_SHORT`, `GL_INT`, and `GL_UNSIGNED_INT`

glVertexAttribPointer also can take: `GL_HALF_FLOAT`, `GL_FLOAT`, `GL_DOUBLE`, `GL_FIXED`, `GL_INT_2_10_10_10_REV`, and `GL_UNSIGNED_INT_2_10_10_10_REV`.

glVertexAttribLPointer takes only `GL_DOUBLE`.

The initial value is `GL_FLOAT`.

normalized

For **glVertexAttribPointer**, specifies whether fixed-point data values should be normalized (`GL_TRUE`) or converted directly as fixed-point values (`GL_FALSE`) when they are accessed.

stride

Specifies the byte offset between consecutive generic vertex attributes. If *stride* is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

pointer

Specifies a offset of the first component of the first generic vertex attribute in the array in the data store of the buffer currently bound to the `GL_ARRAY_BUFFER` target. The initial value is 0.

Description

glVertexAttribPointer, **glVertexAttribIPointer** and **glVertexAttribLPointer** specify the location and data format of the array of generic vertex attributes at index *index* to use when rendering. *size* specifies the number of components per attribute and must be 1, 2, 3, 4, or `GL_BGRA`. *type* specifies the data type of each component, and *stride* specifies the byte stride from one attribute to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.

For **glVertexAttribPointer**, if *normalized* is set to `GL_TRUE`, it indicates that values stored in an integer format are to be mapped to the range [-1,1] (for signed values) or [0,1] (for unsigned values) when they are accessed and converted to floating point. Otherwise, values will be converted to floats directly without normalization.

For **glVertexAttribIPointer**, only the integer types `GL_BYTE`, `GL_UNSIGNED_BYTE`, `GL_SHORT`, `GL_UNSIGNED_SHORT`, `GL_INT`, `GL_UNSIGNED_INT` are accepted. Values are always left as integer values.

glVertexAttribLPointer specifies state for a generic vertex attribute array associated with a shader attribute variable declared with 64-bit double precision components. *type* must be `GL_DOUBLE`. *index*, *size*, and *stride* behave as described for **glVertexAttribPointer** and **glVertexAttribIPointer**.

If *pointer* is not NULL, a non-zero named buffer object must be bound to the `GL_ARRAY_BUFFER` target (see `glBindBuffer`), otherwise an error is generated. *pointer* is treated as a byte offset into the buffer object's data store. The buffer object binding (`GL_ARRAY_BUFFER_BINDING`) is saved as generic vertex attribute array state (`GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING`) for index *index*.

When a generic vertex attribute array is specified, *size*, *type*, *normalized*, *stride*, and *pointer* are saved as vertex array state, in addition to the current vertex array buffer object binding.

To enable and disable a generic vertex attribute array, call `glEnableVertexAttribArray` and `glDisableVertexAttribArray` with *index*. If enabled, the generic vertex attribute array is used when `glDrawArrays`, `glMultiDrawArrays`, `glDrawElements`, `glMultiDrawElements`, OR `glDrawRangeElements` is called.

Notes

Each generic vertex attribute array is initially disabled and isn't accessed when `glDrawElements`, `glDrawRangeElements`, `glDrawArrays`, `glMultiDrawArrays`, OR `glMultiDrawElements` is called.

Errors

`GL_INVALID_VALUE` is generated if *index* is greater than or equal to `GL_MAX_VERTEX_ATTRIBS`.

`GL_INVALID_VALUE` is generated if *size* is not 1, 2, 3, 4 or (for **`glVertexAttribPointer`**), `GL_BGRA`.

`GL_INVALID_ENUM` is generated if *type* is not an accepted value.

`GL_INVALID_VALUE` is generated if *stride* is negative.

`GL_INVALID_OPERATION` is generated if *size* is `GL_BGRA` and *type* is not `GL_INT_2_10_10_10_REV` or `GL_UNSIGNED_INT_2_10_10_10_REV`.

`GL_INVALID_OPERATION` is generated if *type* is `GL_INT_2_10_10_10_REV` or `GL_UNSIGNED_INT_2_10_10_10_REV` and *size* is not 4 or `GL_BGRA`.

`GL_INVALID_OPERATION` is generated by **`glVertexAttribPointer`** if *size* is `GL_BGRA` and *normalized* is `GL_FALSE`.

`GL_INVALID_OPERATION` is generated if zero is bound to the `GL_ARRAY_BUFFER` buffer object binding point and the *pointer* argument is not `NULL`.

Associated Gets

`glGet` with argument `GL_MAX_VERTEX_ATTRIBS`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_ENABLED`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_SIZE`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_TYPE`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_NORMALIZED`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_STRIDE`

`glGetVertexAttrib` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING`

`glGet` with argument `GL_ARRAY_BUFFER_BINDING`

`glGetVertexAttribPointerv` with arguments *index* and `GL_VERTEX_ATTRIB_ARRAY_POINTER`

See Also

`glBindAttribLocation`, `glBindBuffer`, `glDisableVertexAttribArray`, `glDrawArrays`, `glDrawElements`, `glDrawRangeElements`, `glEnableVertexAttribArray`, `glMultiDrawArrays`, `glMultiDrawElements`, `glVertexAttrib`

■ Vertex Array Object

- Vertex Specification

Copyright

Copyright © 2003-2005 3Dlabs Inc. Ltd. This material may be distributed subject to the terms and conditions set forth in the Open Publication License, v 1.0, 8 June 1999.
<http://opencontent.org/openpub/>.

Retrieved from "http://www.opengl.org/wiki_132/index.php?title=GLAPI/glVertexAttribPointer&oldid=6289"

Categories: Core API Ref Vertex Arrays | Core API Reference

- This page was last modified on 29 April 2012, at 10:30.
- This page has been accessed 5,444 times.