Name

glBufferData — creates and initializes a buffer object's data store

C Specification

```
void glBufferData(GLenum         target,
                  GLsizeiptr     size,
                  const GLvoid * data,
                  GLenum         usage);
```

Parameters

target

Specifies the target buffer object. The symbolic constant must be GL_ARRAY_BUFFER, GL_ELEMENT_ARRAY_BUFFER, GL_PIXEL_PACK_BUFFER, or GL_PIXEL_UNPACK_BUFFER.

size

Specifies the size in bytes of the buffer object's new data store.

data

Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

usage

Specifies the expected usage pattern of the data store. The symbolic constant must be GL_STREAM_DRAW, GL_STREAM_READ, GL_STREAM_COPY, GL_STATIC_DRAW, GL_STATIC_READ, GL_STATIC_COPY, GL_DYNAMIC_DRAW, GL_DYNAMIC_READ, or GL_DYNAMIC_COPY.

Description

glBufferData creates a new data store for the buffer object currently bound to *target*. Any pre-existing data store is deleted. The new data store is created with the specified *size* in bytes and *usage*. If *data* is not NULL, the data store is initialized with data from this pointer. In its initial state, the new data store is not mapped, it has a NULL mapped pointer, and its mapped access is GL_READ_WRITE.

usage is a hint to the GL implementation as to how a buffer object's data store will be accessed. This enables the GL implementation to make more intelligent decisions that may significantly impact buffer object performance. It does not, however, constrain the actual usage of the data store. *usage* can be broken down into two parts: first, the frequency of access (modification and usage), and second, the nature of that access. The frequency of access may be one of these:

STREAM

The data store contents will be modified once and used at most a few times.

STATIC

The data store contents will be modified once and used many times.

DYNAMIC

The data store contents will be modified repeatedly and used many times.

The nature of access may be one of these:

DRAW

The data store contents are modified by the application, and used as the source for GL drawing and image specification commands.

READ

The data store contents are modified by reading data from the GL, and used to return that data when queried by the application.

COPY

The data store contents are modified by reading data from the GL, and used as the source for GL drawing and image specification commands.

Notes

`glBufferData` is available only if the GL version is 1.5 or greater.

Targets `GL_PIXEL_PACK_BUFFER` and `GL_PIXEL_UNPACK_BUFFER` are available only if the GL version is 2.1 or greater.

If *`data`* is `NULL`, a data store of the specified size is still created, but its contents remain uninitialized and thus undefined.

Clients must align data elements consistent with the requirements of the client platform, with an additional base-level requirement that an offset within a buffer to a datum comprising *N* bytes be a multiple of *N*.

Errors

`GL_INVALID_ENUM` is generated if *`target`* is not `GL_ARRAY_BUFFER`, `GL_ELEMENT_ARRAY_BUFFER`, `GL_PIXEL_PACK_BUFFER`, or `GL_PIXEL_UNPACK_BUFFER`.

`GL_INVALID_ENUM` is generated if *`usage`* is not `GL_STREAM_DRAW`, `GL_STREAM_READ`, `GL_STREAM_COPY`, `GL_STATIC_DRAW`, `GL_STATIC_READ`, `GL_STATIC_COPY`, `GL_DYNAMIC_DRAW`, `GL_DYNAMIC_READ`, or `GL_DYNAMIC_COPY`.

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

`GL_INVALID_OPERATION` is generated if the reserved buffer object name 0 is bound to *`target`*.

`GL_OUT_OF_MEMORY` is generated if the GL is unable to create a data store with the specified *`size`*.

`GL_INVALID_OPERATION` is generated if `glBufferData` is executed between the execution of glBegin and the corresponding execution of glEnd.

Associated Gets

glGetBufferSubData

glGetBufferParameteriv with argument `GL_BUFFER_SIZE` or `GL_BUFFER_USAGE`

See Also

glBindBuffer, glBufferSubData, glMapBuffer, glUnmapBuffer

Copyright