New 4DSSE Code

0.0.1a

Generated by Doxygen 1.13.2

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 DObject Class Reference

A universal data container class.

`#include <DObject.h>`

Public Types

- using DataType

  Supported data types for the DObject.
- using Plugin = std::function<void(DObject&)>

  Placeholder type for plugins.

Public Member Functions

- DObject ()

  Default constructor.
- DObject (const DataType &data, const Metadata &metadata)

  Constructor to initialize a DObject with data and metadata.
- const DataType & getData () const noexcept

  Retrieves the data stored in the DObject.
- void setData (const DataType &data)

  Sets the data for the DObject.
- const Metadata & getMetadata () const noexcept

  Retrieves the metadata associated with the DObject.
- void setMetadata (const Metadata &metadata)

  Updates the metadata for the DObject.
- void setDebugging (bool enableDebug)

  Enables or disables debugging and tracing for the DObject.
- bool isDebuggingEnabled () const noexcept

  Checks if debugging is enabled for the DObject.
- void registerPlugin (const std::string &id, Plugin plugin)

  Registers a plugin with the DObject.
- void unregisterPlugin (const std::string &id)

  Unregisters a plugin by its identifier.
- void runPlugin (const std::string &id)

  Executes a plugin by its identifier.
- void runAllPlugins ()

  Executes all registered plugins in the registry.

Friends

- std::ostream & operator<< (std::ostream &os, const DObject &obj)

  Provides a human-readable summary of the DObject.

3.1.1 Detailed Description

A universal data container class.

The DObject class is designed to store arbitrary data alongside descriptive metadata. It supports plugin registration to allow extensible functionality.

3.1.2 Member Typedef Documentation

3.1.2.1 DataType

```
using DObject::DataType
```

Initial value:

```
std::variant<
       int, float, double, std::string, std::monostate,
       std::vector<int>, std::vector<float>, std::vector<double>
   >
```

Supported data types for the DObject.

This type alias uses `std::variant` to store different types of data, ensuring type safety and flexibility.

3.1.2.2 Plugin

```
using DObject::Plugin = std::function<void(DObject&)>
```

Placeholder type for plugins.

In the future, this will be replaced with a concrete interface.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 DObject() [1/2]

```
DObject::DObject ()
```

Default constructor.

Creates an empty DObject with default metadata.

3.1.3.2 DObject() [2/2]

```
DObject::DObject (
             const DataType & data,
             const Metadata & metadata)
```

Constructor to initialize a DObject with data and metadata.

Parameters

data	The data to be stored in the DObject.
metadata	Metadata describing the stored data.

3.1.4 Member Function Documentation

3.1.4.1 getData()

```
const DObject::DataType & DObject::getData () const  [noexcept]
```

Retrieves the data stored in the DObject.

Use the appropriate type (matching the stored data) with `std::get<T>()`.

Returns

A constant reference to the stored data.

3.1.4.2 getMetadata()

```
const Metadata & DObject::getMetadata () const  [noexcept]
```

Retrieves the metadata associated with the DObject.

The metadata provides essential information about the stored data, such as its type, size, and dimensions.

Returns

A constant reference to the metadata.

3.1.4.3 isDebuggingEnabled()

```
bool DObject::isDebuggingEnabled () const  [nodiscard], [noexcept]
```

Checks if debugging is enabled for the DObject.

Returns

True if debugging is enabled, false otherwise.

3.1.4.4 registerPlugin()

```
void DObject::registerPlugin (
            const std::string & id,
            Plugin plugin)
```

Registers a plugin with the DObject.

Plugins are stored in a registry and can add custom functionality to the DObject.

Parameters

id	A unique identifier for the plugin.
plugin	The plugin function to register.

3.1.4.5 runAllPlugins()

```
void DObject::runAllPlugins ()
```

Executes all registered plugins in the registry.

Iterates through all plugins and invokes them on the current DObject.

3.1.4.6 runPlugin()

```
void DObject::runPlugin (
            const std::string & id)
```

Executes a plugin by its identifier.

Invokes the registered plugin function. If the plugin is not found, no action is taken.

Parameters

id	The unique identifier of the plugin to execute.

3.1.4.7 setData()

```
void DObject::setData (
            const DataType & data)
```

Sets the data for the DObject.

Updates the stored data and optionally updates metadata.

Parameters

data	The new data to store in the DObject.

3.1.4.8 setDebugging()

```
void DObject::setDebugging (
            bool enableDebug)
```

Enables or disables debugging and tracing for the DObject.

When debugging is enabled, the DObject tracks creation and modification history.

Parameters

enableDebug	True to enable debugging, false to disable it.

3.1.4.9 setMetadata()

```
void DObject::setMetadata (
            const Metadata & metadata)
```

Updates the metadata for the DObject.

Use this function carefully to ensure consistency between the data and metadata.

Parameters

metadata	The new metadata to associate with the DObject.

3.1.4.10 unregisterPlugin()

```
void DObject::unregisterPlugin (
            const std::string & id)
```

Unregisters a plugin by its identifier.

Removes the plugin from the registry if it exists.

Parameters

id	The unique identifier of the plugin to unregister.

3.1.5 Friends And Related Symbol Documentation

3.1.5.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const DObject & obj)  [friend]
```

Provides a human-readable summary of the DObject.

Useful for quick inspection or logging during debugging sessions.

Parameters

os	The output stream to write to.
obj	The DObject to summarize.

Returns

A reference to the output stream.

The documentation for this class was generated from the following files:

- src/dobj/public/DObject.h
- src/dobj/private/DObject.cpp

3.2 LockableDObject Class Reference

Wrapper for DObject with thread-safe access.

`#include <LockableDObject.h>`

Public Member Functions

- **LockableDObject** ()=default
  Default constructor.
- DObject & get ()
  Access the underlying DObject.
- void **lock** ()
  Locks the mutex to ensure thread-safe access.
- void **unlock** ()
  Unlocks the mutex after thread-safe access.

3.2.1 Detailed Description

Wrapper for DObject with thread-safe access.

3.2.2 Member Function Documentation

3.2.2.1 get()

```
DObject & LockableDObject::get ()
```

Access the underlying DObject.

Returns

A reference to the wrapped DObject.

The documentation for this class was generated from the following files:

- src/dobj/public/LockableDObject.h
- src/dobj/private/LockableDObject.cpp

3.3 Metadata Class Reference

Represents metadata information for data objects in the dobj module.

`#include <Metadata.h>`

Public Member Functions

- **Metadata** ()=default

  Default constructor for Metadata.
- Metadata (std::size_t byteSize, std::string dataType, std::vector< std::size_t > dimensions, bool debug↩ Flag=false)

  Constructor to initialize Metadata with specific attributes.
- std::size_t getByteSize () const noexcept

  Gets the total size of the data in bytes.
- void setByteSize (std::size_t byteSize) noexcept

  Sets the total size of the data in bytes.
- const std::string & getDataType () const noexcept

  Gets the type of the data.
- void setDataType (const std::string &dataType)

  Sets the type of the data.
- const std::vector< std::size_t > & getDimensions () const noexcept

  Gets the dimensions of the data.
- void setDimensions (const std::vector< std::size_t > &dimensions)

  Sets the dimensions of the data.
- bool isDebugEnabled () const noexcept

  Checks if debugging information is enabled.
- void setDebugEnabled (bool debugFlag) noexcept

  Sets the debugging flag.

Friends

- std::ostream & operator<< (std::ostream &os, const Metadata &metadata)

  Prints the metadata information for debugging purposes.

3.3.1 Detailed Description

Represents metadata information for data objects in the dobj module.

The Metadata class encapsulates details such as data size, type, dimensions, and optional debugging flags. It is designed to provide descriptive attributes in a lightweight and efficient manner.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Metadata()

```
Metadata::Metadata (
            std::size_t byteSize,
            std::string dataType,
            std::vector< std::size_t > dimensions,
            bool debugFlag = false)
```

Constructor to initialize Metadata with specific attributes.

Parameters

byteSize	The total size of the data in bytes.
dataType	The type of the data (e.g., "float", "double").
dimensions	A vector representing the size of each dimension (e.g., {3, 4} for a 3x4 matrix).
debugFlag	Whether debugging information is enabled for this Metadata instance.

3.3.3 Member Function Documentation

3.3.3.1 getByteSize()

```
std::size_t Metadata::getByteSize () const  [nodiscard], [noexcept]
```

Gets the total size of the data in bytes.

Returns

The total byte size of the data.

The size is often required for memory allocation and validation in numerical routines.

Returns

The total byte size of the data.

3.3.3.2 getDataType()

```
const std::string & Metadata::getDataType () const  [nodiscard], [noexcept]
```

Gets the type of the data.

Returns

A string representing the data type.

The type (e.g., "float", "double") is critical for casting raw data or interfacing with libraries that require specific types.

Returns

A string representing the data type.

3.3.3.3 getDimensions()

```
const std::vector< std::size_t > & Metadata::getDimensions () const  [nodiscard], [noexcept]
```

Gets the dimensions of the data.

Returns

A vector representing the size of each dimension.

Dimensions define the shape of the data (e.g., 2D arrays, 3D matrices). This is essential for ensuring that operations (e.g., matrix multiplication) are performed correctly.

Returns

A vector representing the size of each dimension.

3.3.3.4 isDebugEnabled()

```
bool Metadata::isDebugEnabled () const  [nodiscard], [noexcept]
```

Checks if debugging information is enabled.

Returns

True if debugging is enabled, false otherwise.

Debugging flags can be useful for tracking performance metrics or error provenance.

Returns

True if debugging is enabled, false otherwise.

3.3.3.5 setByteSize()

```
void Metadata::setByteSize (
            std::size_t byteSize)  [noexcept]
```

Sets the total size of the data in bytes.

Parameters

byteSize	The total byte size to set.

It's important to ensure this matches the actual data size in memory to prevent overflows or incorrect computations downstream.

Parameters

byteSize	The total byte size to set.

3.3.3.6 setDataType()

```
void Metadata::setDataType (
            const std::string & dataType)
```

Sets the type of the data.

Parameters

dataType	A string representing the data type.

When setting the data type, ensure it aligns with the underlying data representation. Mismatched types can lead to undefined behavior in numerical calculations.

Parameters

dataType	A string representing the data type.

3.3.3.7 setDebugEnabled()

```
void Metadata::setDebugEnabled (
            bool debugFlag)  [noexcept]
```

Sets the debugging flag.

Parameters

debugFlag	Whether debugging is enabled.

Enabling debugging can introduce performance overhead but provides valuable insights during development or testing.

Parameters

debugFlag	Whether debugging is enabled.

3.3.3.8 setDimensions()

```
void Metadata::setDimensions (
            const std::vector< std::size_t > & dimensions)
```

Sets the dimensions of the data.

Parameters

dimensions	A vector representing the size of each dimension.

When modifying dimensions, verify that they are consistent with the actual data representation.

Parameters

dimensions	A vector representing the size of each dimension.

3.3.4 Friends And Related Symbol Documentation

3.3.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Metadata & metadata)  [friend]
```

Prints the metadata information for debugging purposes.

Parameters

os	The output stream to print to.
metadata	The Metadata object to print.

Returns

A reference to the output stream.

This function provides a human-readable summary of the metadata. Useful for quick checks or logging during debugging sessions.

Parameters

os	The output stream to print to.
metadata	The Metadata object to print.

Returns

A reference to the output stream.

The documentation for this class was generated from the following files:

- src/dobj/public/Metadata.h
- src/dobj/private/Metadata.cpp

Chapter 4

File Documentation

4.1 DebugInfo.h

```
00001
```

4.2 src/dobj/private/DObject.cpp File Reference

Implementation of the DObject class.

```
#include "DObject.h"
#include <iostream>
#include <stdexcept>
```

4.2.1 Detailed Description

Implementation of the DObject class.

Provides the implementation for a universal data container with plugin support.

4.3 DObjectPrivate.h

```
00001
```

4.4 src/dobj/private/Metadata.cpp File Reference

Implementation of the Metadata class used in the dobj module.

```
#include "Metadata.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const Metadata &metadata)

  Prints the metadata information for debugging purposes.

4.4.1 Detailed Description

Implementation of the Metadata class used in the dobj module.

Provides methods to manage metadata for data objects, including size, type, dimensions, and debugging flags.

4.4.2 Function Documentation

4.4.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Metadata & metadata)
```

Prints the metadata information for debugging purposes.

This function provides a human-readable summary of the metadata. Useful for quick checks or logging during debugging sessions.

Parameters

os	The output stream to print to.
metadata	The Metadata object to print.

Returns

A reference to the output stream.

4.5 src/dobj/public/DObject.h File Reference

Defines the DObject class, a universal data container for the project.

```
#include "Metadata.h"
#include <variant>
#include <memory>
#include <vector>
#include <string>
#include <mutex>
#include <map>
#include <functional>
```

Classes

- class DObject

  A universal data container class.

4.5.1 Detailed Description

Defines the DObject class, a universal data container for the project.

The DObject class encapsulates arbitrary data and its associated metadata, providing a consistent interface for public-facing functions. It includes support for dynamically registered plugins.

4.6 DObject.h

Go to the documentation of this file.

```
00001 #ifndef DOBJECT_H
00002 #define DOBJECT_H
00003
00004 #include "Metadata.h"
00005 #include <variant>
00006 #include <memory>
00007 #include <vector>
00008 #include <string>
00009 #include <mutex>
00010 #include <map>
00011 #include <functional>
00012
00021
00029 class DObject {
00030 public:
00037     using DataType = std::variant<
00038         int, float, double, std::string, std::monostate,
00039         std::vector<int>, std::vector<float>, std::vector<double>
00040     >;
00041
00047     using Plugin = std::function<void(DObject&)>;
00048
00054     DObject();
00055
00062     DObject(const DataType& data, const Metadata& metadata);
00063
00071     const DataType& getData() const noexcept;
00072
00080     void setData(const DataType& data);
00081
00090     const Metadata& getMetadata() const noexcept;
00091
00099     void setMetadata(const Metadata& metadata);
00100
00108     void setDebugging(bool enableDebug);
00109
00115     [[nodiscard]] bool isDebuggingEnabled() const noexcept;
00116
00125     void registerPlugin(const std::string& id, Plugin plugin);
00126
00134     void unregisterPlugin(const std::string& id);
00135
00143     void runPlugin(const std::string& id);
00144
00150     void runAllPlugins();
00151
00161     friend std::ostream& operator<<(std::ostream& os, const DObject& obj);
00162
00163 private:
00164     DataType data_;
00165     Metadata metadata_;
00166     bool debugEnabled_ = false;
00167     std::map<std::string, Plugin> plugins_;
00168 };
00169
00170 #endif // DOBJECT_H
```

4.7 src/dobj/public/LockableDObject.h File Reference

A lightweight wrapper for DObject that adds locking capabilities.

```
#include "DObject.h"
#include <mutex>
```

Classes

- class LockableDObject

  Wrapper for DObject with thread-safe access.

4.7.1 Detailed Description

A lightweight wrapper for DObject that adds locking capabilities.

This class allows safe concurrent access to a DObject by providing locking and unlocking methods.

4.8 LockableDObject.h

Go to the documentation of this file.

```
00001 #ifndef LOCKABLE_DOBJECT_H
00002 #define LOCKABLE_DOBJECT_H
00003
00004 #include "DObject.h"
00005 #include <mutex>
00006
00014
00019 class LockableDObject {
00020 public:
00024     LockableDObject() = default;
00025
00030     DObject& get();
00031
00035     void lock();
00036
00040     void unlock();
00041
00042 private:
00043     DObject object_;
00044     std::mutex mutex_;
00045 };
00046
00047 #endif // LOCKABLE_DOBJECT_H
```

4.9 src/dobj/public/Metadata.h File Reference

Public interface for the Metadata class used in the dobj module.

```
#include <string>
#include <vector>
#include <cstddef>
#include <iostream>
```

Classes

- class Metadata

  Represents metadata information for data objects in the dobj module.

4.9.1 Detailed Description

Public interface for the Metadata class used in the dobj module.

The Metadata class provides descriptive information about the data encapsulated within a dobj, including size, type, dimensions, and debugging flags.

4.10 Metadata.h

Go to the documentation of this file.

```
00001 #ifndef METADATA_H
00002 #define METADATA_H
00003
00004 #if defined(__APPLE__) || defined(__linux__)
00005 #define EXPORT_SYMBOL __attribute__((visibility("default")))
00006 #else
00007 #define EXPORT_SYMBOL
00008 #endif
00009
00010 #include <string>
00011 #include <vector>
00012 #include <cstddef>
00013 #include <iostream>
00014
00022
00031 class EXPORT_SYMBOL Metadata {
00032 public:
00036     Metadata() = default;
00037
00046     Metadata(std::size_t byteSize, std::string dataType, std::vector<std::size_t> dimensions, bool
      debugFlag = false);
00047
00052     [[nodiscard]] std::size_t getByteSize() const noexcept;
00053
00058     void setByteSize(std::size_t byteSize) noexcept;
00059
00064     [[nodiscard]] const std::string& getDataType() const noexcept;
00065
00070     void setDataType(const std::string& dataType);
00071
00076     [[nodiscard]] const std::vector<std::size_t>& getDimensions() const noexcept;
00077
00082     void setDimensions(const std::vector<std::size_t>& dimensions);
00083
00088     [[nodiscard]] bool isDebugEnabled() const noexcept;
00089
00094     void setDebugEnabled(bool debugFlag) noexcept;
00095
00103     friend std::ostream& operator<<(std::ostream& os, const Metadata& metadata);
00104
00105 private:
00106     std::size_t byteSize_ = 0;
00107     std::string dataType_;
00108     std::vector<std::size_t> dimensions_;
00109     bool debugFlag_ = false;
00110 };
00111
00112 #endif // METADATA_H
```

Index