MindEngine

Generated by Doxygen 1.8.11

Contents

Chapter 1

Todo List

Member Database::DataNode::identifier

Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

Member Rules::Rule::action ()=0

Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 Database Namespace Reference

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent.

Classes

- class DataGroup

  Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only.
- class DataNode

  Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.
- class Datum

  A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

5.1.1 Detailed Description

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent.

5.2 Rules Namespace Reference

Contains the implementation of rules as well as the mechanism to match the rules and the data in the database.

Classes

- struct Match

  Provides the mechanism to match the data item from the rule with any item inside the database.
- class Rule

  Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

5.2.1 Detailed Description

Contains the implementation of rules as well as the mechanism to match the rules and the data in the database.

Chapter 6

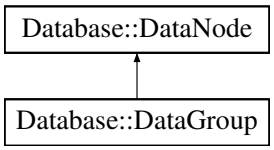
Class Documentation

6.1 Database::DataGroup Class Reference

Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only.

`#include <DataGroup.h>`

Inheritance diagram for Database::DataGroup:



Public Member Functions

- DataGroup ()

  DataGroup default constructor.
- DataGroup (const std::string &identifier, DataNode *parent, DataNode *rightSibling, DataNode *leftMost↩ Child)

  DataGroup constructor.
- virtual ~DataGroup ()

  ~DataGroup destructor.
- const DataNode * getLeftMostChild ()

  Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.
- bool isGroup ()

  Allows user to check whether this node is a DataGroup or not.

Additional Inherited Members

6.1.1 Detailed Description

Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only.

See also

DataNode
Datum

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Database::DataGroup::DataGroup (const std::string & *identifier*, DataNode * *parent*, DataNode * *rightSibling*, DataNode * *leftMostChild*)

DataGroup constructor.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree.
leftMostChild	a DataNode pointer representing this node's left most child in tree.

6.1.3 Member Function Documentation

6.1.3.1 const DataNode * Database::DataGroup::getLeftMostChild ( )

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.

Returns

The left most child of this data group node.

6.1.3.2 bool Database::DataGroup::isGroup ( ) [virtual]

Allows user to check whether this node is a DataGroup or not.

Returns

true if this node is a DataGroup, otherwise returns false.

Reimplemented from Database::DataNode.

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

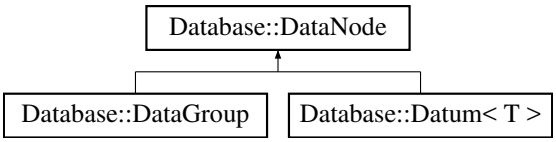
- Engine/Database/DataGroup.h
- Engine/Database/DataGroup.cpp

6.2 Database::DataNode Class Reference

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

`#include <DataNode.h>`

Inheritance diagram for Database::DataNode:



Public Member Functions

- DataNode ()
  DataNode default constructor.
- DataNode (const std::string &identifier, DataNode *parent, DataNode *rightSibling)
  DataNode constructor with parameters.
- virtual ~DataNode ()
  ~DataNode destructor.
- const std::string & getIdentifier ()
  The data nodes have unique identifiers.
- const DataNode * getParent ()
  Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the parent of this node.
- const DataNode * getRightSibling ()
  Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.
- virtual bool isGroup ()
  Allows user to check whether this node is a DataGroup or not.
- virtual bool isDatum ()
  Allows user to check whether this node is a Datum or not.

Protected Attributes

- std::string identifier
  Each item of data should be identified to know what the value means.
- DataNode * parent
  The parent node of this node, or NULL if this node is the root of the tree.
- DataNode * rightSibling
  The right sibling node of this node, or NULL if this node is the right most child.

6.2.1 Detailed Description

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

See also

DataGroup
Datum

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Database::DataNode::DataNode (const std::string & *identifier*, DataNode * *parent*, DataNode * *rightSibling*)

DataNode constructor with parameters.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree. structure

6.2.3 Member Function Documentation

6.2.3.1 const std::string & Database::DataNode::getIdentifier ( )

The data nodes have unique identifiers.

Returns

The identifier of this node.

6.2.3.2 const DataNode * Database::DataNode::getParent ( )

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the parent of this node.

Returns

The parent node of this node, or NULL if this node is the root of the tree.

6.2.3.3 const DataNode * Database::DataNode::getRightSibling ( )

Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.

Returns

The right sibling node of this node, or NULL if this node is the right most child.

6.2.3.4 bool Database::DataNode::isDatum ( ) [virtual]

Allows user to check whether this node is a Datum or not.

Returns

true if this node is a Datum, otherwise returns false.

Reimplemented in Database::Datum< T >.

6.2.3.5 bool Database::DataNode::isGroup ( ) [virtual]

Allows user to check whether this node is a DataGroup or not.

Returns

true if this node is a DataGroup, otherwise returns false.

Reimplemented in Database::DataGroup.

6.2.4 Member Data Documentation

6.2.4.1 std::string Database::DataNode::identifier [protected]

Each item of data should be identified to know what the value means.

Todo Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

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

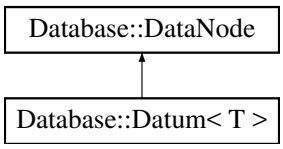
- Engine/Database/DataNode.h
- Engine/Database/DataNode.cpp

6.3 Database::Datum< T > Class Template Reference

A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

`#include <Datum.h>`

Inheritance diagram for Database::Datum< T >:



Public Member Functions

- Datum (T value)
  Datum constructor.
- Datum (const std::string &identifier, DataNode *parent, DataNode *rightSibling, T value)
  Datum constructor.
- virtual ~Datum ()
  ~Datum destructor
- void setValue (T newValue)
  To change the value of the Datum.
- T getValue ()
  To get the value of the Datum.
- bool isDatum ()
  Allows user to check whether this node is a Datum or not.

Additional Inherited Members

6.3.1 Detailed Description

template<class T>
class Database::Datum< T >

A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

See also

DataNode
DataGroup

6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<class T > Database::Datum< T >::Datum (T *value*)

Datum constructor.

Parameters

value	the value that the Datum holds.

6.3.2.2 template<class T > Database::Datum< T >::Datum (const std::string & *identifier*, DataNode * *parent*, DataNode * *rightSibling*, T *value*)

Datum constructor.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree.
value	the value that the Datum holds.

6.3.3 Member Function Documentation

6.3.3.1 template<class T > T Database::Datum< T >::getValue ( )

To get the value of the Datum.

Returns

The value that the datum is currently holding.

6.3.3.2 template<class T > bool Database::Datum< T >::isDatum ( ) [virtual]

Allows user to check whether this node is a Datum or not.

Returns

true if this node is a Datum, otherwise returns false.

Reimplemented from Database::DataNode.

6.3.3.3 template<class T > void Database::Datum< T >::setValue (T *newValue*)

To change the value of the Datum.

Parameters

newValue	The new value that is going to be assigned to the Datum.

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

- Engine/Database/Datum.h
- Engine/Database/Datum.cpp

6.4 Rules::Match Struct Reference

Provides the mechanism to match the data item from the rule with any item inside the database.

`#include <Match.h>`

Public Member Functions

- virtual bool matches (const Database::DataNode *database, void *bindings)=0
  Check the match on the database.

6.4.1 Detailed Description

Provides the mechanism to match the data item from the rule with any item inside the database.

6.4.2 Member Function Documentation

6.4.2.1 virtual bool Rules::Match::matches (const Database::DataNode * *database*, void * *bindings*) [pure virtual]

Check the match on the database.

Parameters

database	The database to match on.
bindings	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both input and output parameter.

Returns

true if matches, else returns false.

The documentation for this struct was generated from the following file:

- Engine/Rules/Match.h

6.5 Rules::Rule Class Reference

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

`#include <Rule.h>`

Public Member Functions

- virtual void action ()=0

  The action is going to be carried out when the rule matches.

Public Attributes

- Match * ifClause

  Consist of a set of data items, in a similar format to those in the database.

6.5.1 Detailed Description

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

6.5.2 Member Function Documentation

6.5.2.1 virtual void Rules::Rule::action ( ) [pure virtual]

The action is going to be carried out when the rule matches.

Todo Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

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

- Engine/Rules/Rule.h

Index