Repast HPC::ReLogo

2.0

Generated by Doxygen 1.8.4

Sun Aug 11 2013 13:43:51

Contents

Chapter 1

Repast HPC- ReLogo Logo-Like Semantics for Repast HPC

By Argonne National Laboratory, 2009-2013

1.1 What is ReLogo

ReLogo is a collection of classes and methods that allow Repast HPC simulations (agent-based simulations for high-performance computing environments) to be built using simple and easily apprehensible semantics.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

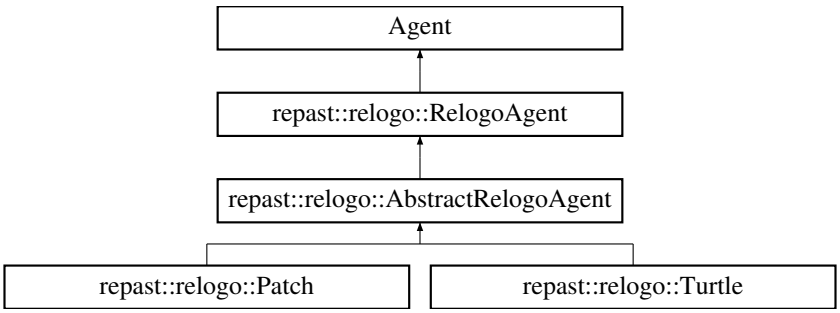
Class Documentation

4.1 repast::relogo::AbstractRelogoAgent Class Reference

Abstract base class for turtles and patches.

`#include <AbstractRelogoAgent.h>`

Inheritance diagram for repast::relogo::AbstractRelogoAgent:



Public Member Functions

- **AbstractRelogoAgent** (AgentId id, Observer *observer)
- virtual int pxCor () const =0

  Gets the patch x coordinate of the agent's location.
- virtual int pyCor () const =0

  Gets the patch y coordinate of the agent's location.
- template<typename AgentType >
  void inRadius (AgentSet< RelogoAgent > &inSet, double radius, AgentSet< AgentType > &outSet) const

  Gets all the agents in the inSet within the specified radius for this RelogoAgent and put them in the outSet.
- template<typename PatchType >
  PatchType * patchAt (double dx, double dy) const

  Gets the patch at direction dx, dy from the this agent.
- template<typename AgentType >
  void turtlesHere (AgentSet< AgentType > &set) const

  Gets all the turtles on this turtle's patch and puts them into the specifed set.
- template<typename AgentType >
  AgentSet< AgentType > turtlesHere () const

  Gets all the turtles on this turtle's patch and returns them in an AgentSet.

- template<typename PatchType >
  PatchType * patchAtHeadingAndDistance (float heading, double distance)

  Gets the patch at the specified heading and distance from this patch or turtle.
- template<typename AgentType >
  void turtlesOn (AgentSet< AgentType > &out) const

  Gets the turtles that are on this patch or if this is a Turtle get the turtles that are on the patch this turtle is on.
- template<typename AgentType >
  AgentSet< AgentType > turtlesOn () const

  Gets the turtles that are on this patch or if this is a Turtle get the turtles that are on the patch this turtle is on.

4.1.1 Detailed Description

Abstract base class for turtles and patches.

This contains some methods that can apply to either.

4.1.2 Member Function Documentation

4.1.2.1 template<typename AgentType > void repast::relogo::AbstractRelogoAgent::inRadius (AgentSet< RelogoAgent > & *inSet*, double *radius*, AgentSet< AgentType > & *outSet*) const

Gets all the agents in the inSet within the specified radius for this RelogoAgent and put them in the outSet.

Parameters

inSet	the set of agents to test if they are withinthe radius

Template Parameters

the	type of agents to include in the outSet

4.1.2.2 template<typename PatchType > PatchType * repast::relogo::AbstractRelogoAgent::patchAt (double *dx*, double *dy*) const

Gets the patch at direction dx, dy from the this agent.

If the resulting location is outside of the world, this returns 0.

Parameters

dx	the distance from the caller along the x dimension
dy	the distance from the caller along the y dimension

Template Parameters

the	type of the Patch

Returns

the patch at that distance from this Turtle, or 0 if the resulting location is outside of the world.

4.1.2.3 template<typename PatchType > PatchType * repast::relogo::AbstractRelogoAgent::patchAtHeadingAndDistance (float *heading*, double *distance*)

Gets the patch at the specified heading and distance from this patch or turtle.

Parameters

heading	the heading
distance	the distance

Template Parameters

PatchType	the patch's type

4.1.2.4 virtual int repast::relogo::AbstractRelogoAgent::pxCor ( ) const [pure virtual]

Gets the patch x coordinate of the agent's location.

Returns

the patch x coordinate

Implements repast::relogo::RelogoAgent.

Implemented in repast::relogo::Turtle, and repast::relogo::Patch.

4.1.2.5 virtual int repast::relogo::AbstractRelogoAgent::pyCor ( ) const [pure virtual]

Gets the patch y coordinate of the agent's location.

Returns

the patch y coordinate

Implements repast::relogo::RelogoAgent.

Implemented in repast::relogo::Turtle, and repast::relogo::Patch.

4.1.2.6 template<typename AgentType > void repast::relogo::AbstractRelogoAgent::turtlesHere (AgentSet< AgentType > & *set*) const

Gets all the turtles on this turtle's patch and puts them into the specifed set.

Parameters

set	the set to put the found turtles in

Template Parameters

AgentType	the type of turtles to get

4.1.2.7 template<typename AgentType > AgentSet< AgentType > repast::relogo::AbstractRelogoAgent::turtlesHere ( ) const

Gets all the turtles on this turtle's patch and returns them in an AgentSet.

Template Parameters

AgentType	the type of turtles to get

Returns

an AgentSet containing all the turtles on this turtles patch.

4.1.2.8 template<typename AgentType > void repast::relogo::AbstractRelogoAgent::turtlesOn (AgentSet< AgentType > & *out*) const

Gets the turtles that are on this patch or if this is a Turtle get the turtles that are on the patch this turtle is on.

Template Parameters

AgentType	the type of turtle

Parameters

out	the turtles will be put in out

4.1.2.9 template<typename AgentType > AgentSet< AgentType > repast::relogo::AbstractRelogoAgent::turtlesOn ( ) const

Gets the turtles that are on this patch or if this is a Turtle get the turtles that are on the patch this turtle is on.

Template Parameters

AgentType	the type of turtle

Returns

an AgentSet containing all the turtles that are on the patch the caller is on.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/AbstractRelogoAgent.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/AbstractRelogoAgent.cpp

4.2 repast::relogo::AgentSet< T > Class Template Reference

Specialized indexable collection class for agents.

`#include <AgentSet.h>`

Public Types

- typedef std::vector< T * > ::iterator **as_iterator**
- typedef std::vector< T * > ::const_iterator **const_as_iterator**

Public Member Functions

- AgentSet ()

  Creates an empty agent set.
- template<typename input_iterator > AgentSet (input_iterator start, input_iterator end)

  Creates an agent set and fills with elements from start through end.
- AgentSet (const AgentSet &set)

  Copy constructor.
- template<typename input_iterator > void addAll (input_iterator begin, input_iterator end)

  Adds all the agents from the start iterator through the end to this AgentSet.
- void add (T *agent)

  Adds an agent to this AgentSet.

- template<typename Functor >
  void ask (Functor func)

  Calls the Functor on each agent in this AgentSet.
- template<typename Functor , typename P1 >
  void ask (Functor func, const P1 &p1)

  Calls the Functor on each agent in this AgentSet, passing the specified argument.
- template<typename Functor , typename P1 >
  void ask (Functor func, P1 &p1)

  Calls the Functor on each agent in this AgentSet, passing the specified argument.
- template<typename Functor >
  void apply (Functor &func)

  Applies the functor to each each agent in the agent set.
- template<typename Functor >
  void apply (const Functor &func)

  Applies the functor to each each agent in the agent set.
- T * at (int index)

  Gets the item at the specified index.
- size_t count () const

  Gets the size of this AgentSet.
- size_t size () const

  Gets the size of this AgentSet.
- T * operator[] (size_t index)

  Gets the item at the specified index without doing any range checking.
- as_iterator begin ()

  Gets an iterator to the begining of this AgentSet.
- const_as_iterator begin () const

  Gets a const iterator to the begining of this AgentSet.
- as_iterator end ()

  Gets an iterator to the end of this AgentSet.
- const_as_iterator end () const

  Gets a const iterator to the end of this AgentSet.
- void clear ()

  Clears this AgentSet of any agents that it contains.
- template<typename ValueGetter >
  T * minOneOf (const ValueGetter &getter)

  Gets the set member that has the minimum value of the number returned by ValueGetter.
- template<typename ValueGetter >
  T * maxOneOf (const ValueGetter &getter)

  Gets the set member that has the maximum value of the number returned by ValueGetter.
- template<typename ValueGetter >
  void withMin (const ValueGetter &getter, AgentSet< T > &set)

  Gets the set members that have the minimum value of the number returned by ValueGetter, and puts them in the specified set.
- template<typename ValueGetter >
  void withMax (const ValueGetter &getter, AgentSet< T > &set)

  Gets the set members that have the maximum value of the number returned by ValueGetter and puts them in the specified set.
- template<typename ValueGetter >
  void minNOf (size_t count, const ValueGetter &getter, AgentSet< T > &set, bool initialSetIsSorted=false)

  Gets count number of set members that have the minimum value of the number returned by ValueGetter.
- template<typename ValueGetter >
  void maxNOf (size_t count, const ValueGetter &getter, AgentSet< T > &set, bool initialSetIsSorted=false)

Gets count number of set members that have the maximum value of the number returned by ValueGetter.

- T * oneOf ()

  Gets one of the members of this AgentSet at random.

- void remove (T *agent)

  Removes all instances of the specified agent from this AgentSet.

- void shuffle ()

  Randomly shuffles the elements of this AgentSet.

Public Attributes

- std::vector< T * > **agents**

4.2.1 Detailed Description

template<typename T>class repast::relogo::AgentSet< T >

Specialized indexable collection class for agents.

This includes methods designed to call arbitrary code on the agents it contains.

Template Parameters

T	the type of agent the AgentSet contains

4.2.2 Member Function Documentation

4.2.2.1 template<typename T > template<typename Functor > void repast::relogo::AgentSet< T >::apply (Functor & *func*)

Applies the functor to each each agent in the agent set.

Template Parameters

Functor	an object that implements operator()(T* agent);

4.2.2.2 template<typename T > template<typename Functor > void repast::relogo::AgentSet< T >::apply (const Functor & *func*)

Applies the functor to each each agent in the agent set.

Template Parameters

Functor	an object that implements operator()(T* agent);

4.2.2.3 template<typename T > template<typename Functor > void repast::relogo::AgentSet< T >::ask (Functor *func*)

Calls the Functor on each agent in this AgentSet.

Parameters

func	a pointer to the method to call each member of the set

Template Parameters

Functor	pointer to no-arg method belonging to the type of agent contained by this AgentSet.

4.2.2.4 template<typename T > template<typename Functor , typename P1 > void repast::relogo::AgentSet< T >::ask (Functor *func*, const P1 & *p1*)

Calls the Functor on each agent in this AgentSet, passing the specified argument.

Parameters

func	a pointer to the method to call each member of the set
p1	a reference to a P1 type that is passed to the called method

Template Parameters

Functor	pointer to method belonging to the type of agent contained by this AgentSet.
P1	the type of the method parameter

4.2.2.5 template<typename T > template<typename Functor , typename P1 > void repast::relogo::AgentSet< T >::ask (Functor *func*, P1 & *p1*)

Calls the Functor on each agent in this AgentSet, passing the specified argument.

Parameters

func	a pointer to the method to call each member of the set
p1	a reference to a P1 type that is passed to the called method

Template Parameters

Functor	pointer to method belonging to the type of agent contained by this AgentSet.
P1	the type of the method parameter

4.2.2.6 template<typename T > T * repast::relogo::AgentSet< T >::at (int *index*)

Gets the item at the specified index.

Parameters

index	the index of the agent to get

Returns

the agent at the specified index

4.2.2.7 template<typename T> as_iterator repast::relogo::AgentSet< T >::begin ( ) [inline]

Gets an iterator to the begining of this AgentSet.

Returns

an iterator to the beginning of this AgentSet.

4.2.2.8 template<typename T> const_as_iterator repast::relogo::AgentSet< T >::begin ( ) const [inline]

Gets a const iterator to the begining of this AgentSet.

Returns

a const iterator to the beginning of this AgentSet.

4.2.2.9 template<typename T> size_t repast::relogo::AgentSet< T >::count ( ) const [inline]

Gets the size of this AgentSet.

Returns

the size of this AgentSet.

4.2.2.10 template<typename T> as_iterator repast::relogo::AgentSet< T >::end ( ) [inline]

Gets an iterator to the end of this AgentSet.

Returns

an iterator to the end of this AgentSet.

4.2.2.11 template<typename T> const_as_iterator repast::relogo::AgentSet< T >::end ( ) const [inline]

Gets a const iterator to the end of this AgentSet.

Returns

a const iterator to the end of this AgentSet.

4.2.2.12 template<typename T > template<typename ValueGetter > void repast::relogo::AgentSet< T >::maxNOf (size_t *count,* const ValueGetter & *getter,* AgentSet< T > & *set,* bool *initialSetIsSorted* = false)

Gets count number of set members that have the maximum value of the number returned by ValueGetter.

If there are not enough to satisfy the count then members with the second lowest value are returned and so on.

Parameters

getter	the ValueGetter to use in retreiving the value used in the max comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the max comparison.

Parameters

initialSetIsSorted	Optional performance parameter; if false (the default), a call to this function must sort an entire copy of the original set; if true, the function assumes the original set is already sorted. Useful if the same set is to be used repeatedly.

4.2.2.13 template<typename T > template<typename ValueGetter > T * repast::relogo::AgentSet< T >::maxOneOf (const ValueGetter & *getter*)

Gets the set member that has the maximum value of the number returned by ValueGetter.

If more than one agent has the minimum value, then return one of those at random.

Parameters

getter	the ValueGetter to use in retreiving the value used in the max comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the max comparison.

4.2.2.14 template<typename T > template<typename ValueGetter > void repast::relogo::AgentSet< T >::minNOf (size_t *count*, const ValueGetter & *getter*, AgentSet< T > & *set*, bool *initialSetIsSorted* = `false`)

Gets count number of set members that have the minimum value of the number returned by ValueGetter.

If there are not enough to satisfy the count then members with the second lowest value are returned and so on.

Parameters

getter	the ValueGetter to use in retreiving the value used in the min comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the min comparison.

Parameters

initialSetIsSorted	Optional performance parameter; if false (the default), a call to this function must sort an entire copy of the original set; if true, the function assumes the original set is already sorted. Useful if the same set is to be used repeatedly.

4.2.2.15 template<typename T > template<typename ValueGetter > T * repast::relogo::AgentSet< T >::minOneOf (const ValueGetter & *getter*)

Gets the set member that has the minimum value of the number returned by ValueGetter.

If more than one agent has the minimum value, then return one of those at random.

Parameters

getter	the ValueGetter to use in retreiving the value used in the min comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the min comparison.

4.2.2.16 template<typename T > T * repast::relogo::AgentSet< T >::oneOf ( )

Gets one of the members of this AgentSet at random.

If the set is empty, this returns 0.

4.2.2.17 template<typename T > T * repast::relogo::AgentSet< T >::operator[] (size_t *index*)

Gets the item at the specified index without doing any range checking.

Parameters

index	the index of the agent to get

Returns

the agent at the specified index

4.2.2.18 template<typename T> size_t repast::relogo::AgentSet< T >::size ( ) const [inline]

Gets the size of this AgentSet.

Returns

the size of this AgentSet.

4.2.2.19 template<typename T > template<typename ValueGetter > void repast::relogo::AgentSet< T >::withMax (const ValueGetter & *getter*, AgentSet< T > & *set*)

Gets the set members that have the maximum value of the number returned by ValueGetter and puts them in the specified set.

Parameters

getter	the ValueGetter to use in retreiving the value used in the max comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the max comparison.

4.2.2.20 template<typename T > template<typename ValueGetter > void repast::relogo::AgentSet< T >::withMin (const ValueGetter & *getter*, AgentSet< T > & *set*)

Gets the set members that have the minimum value of the number returned by ValueGetter, and puts them in the specified set.

Parameters

getter	the ValueGetter to use in retreiving the value used in the min comparison

Template Parameters

ValueGetter	a function or functor that takes a member of this agentset and returns a double value. This double value is used in the min comparison.

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

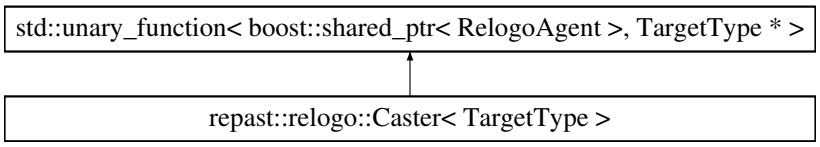
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/AgentSet.h

4.3 repast::relogo::Caster< TargetType > Struct Template Reference

Unary function used in the transform_iterator that allows context iterators to return the agent maps values.

`#include <Observer.h>`

Inheritance diagram for repast::relogo::Caster< TargetType >:



Public Member Functions

- TargetType * **operator()** (boost::shared_ptr< RelogoAgent > ptr) const

4.3.1 Detailed Description

template<typename TargetType>struct repast::relogo::Caster< TargetType >

Unary function used in the transform_iterator that allows context iterators to return the agent maps values.

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

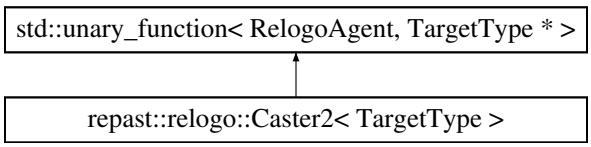
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Observer.h

4.4 repast::relogo::Caster2< TargetType > Struct Template Reference

Unary function used in the transform_iterator that allows.

`#include <agent_set_functions.h>`

Inheritance diagram for repast::relogo::Caster2< TargetType >:



Public Member Functions

- TargetType * **operator()** (const RelogoAgent *agent) const

4.4.1 Detailed Description

template<typename TargetType>struct repast::relogo::Caster2< TargetType >

Unary function used in the transform_iterator that allows.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/agent_set_functions.h

4.5 repast::relogo::DefaultAgentCreator< Agent > Struct Template Reference

operator() that creates an agent of type Agent.

`#include <creators.h>`

Public Member Functions

- Agent * **operator()** (const repast::AgentId &id, Observer *obs)

4.5.1 Detailed Description

template<typename Agent>struct repast::relogo::DefaultAgentCreator< Agent >

operator() that creates an agent of type Agent.

The type Agent must have a constructor that takes an AgentId and pointer to an Observer.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/creators.h

4.6 repast::relogo::DefaultLinkCreator Struct Reference

operator() that creates a RelogoLink from a source and target RelogoAgents.

`#include <creators.h>`

Public Member Functions

- RelogoLink * **operator()** (RelogoAgent *source, RelogoAgent *target)

4.6.1 Detailed Description

operator() that creates a RelogoLink from a source and target RelogoAgents.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/creators.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/creators.cpp

4.7 repast::relogo::IsAgentTypeNoDup< T > Struct Template Reference

Used to filter by agent type but ensure that only the first encountered instance of agent is considered.

`#include <agent_set_functions.h>`

Public Member Functions

- **IsAgentTypeNoDup** (int typeId)
- bool **operator()** (const T *agent)

Public Attributes

- IsAgentType< T > **isAgentType**
- boost::unordered_set< AgentId, HashId > **set**

4.7.1 Detailed Description

template<typename T>struct repast::relogo::IsAgentTypeNoDup< T >

Used to filter by agent type but ensure that only the first encountered instance of agent is considered.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/agent_set_functions.h

4.8 repast::relogo::Observer Class Reference

Implementation of a logo Observer.

`#include <Observer.h>`

Public Member Functions

- void addDataSet (repast::DataSet *dataSet)

  Adds a dataset to this Observer.
- void dataSetClose ()

  Non API method for closing all the datasets at the end of a sim runs.
- virtual void go ()=0

  Called every tick of the simulation.
- virtual void setup (Properties &props)

  Classes that extend this should include model initialization here.
- void _setup (Properties &props)

  Performs internal Relogo initialization.
- int minPxcor () const

  Gets the minimum x coordinate of the patches managed by this Observer.
- int minPycor () const

  Gets the minimum y coordinate of the patches managed by this Observer.
- int maxPxcor () const

  Gets the maximum x coordinate of the patches managed by this Observer.
- int maxPycor () const

  Gets the maximum y coordinate of the patches managed by this Observer.
- int randomPxcor ()

  Gets a random x coodinate of the patches managed by this Observer.
- int randomPycor ()

  Gets a random y coodinate of the patches managed by this Observer.
- double randomXcor ()

  Gets a random x coodinate of the turtles managed by this Observer.
- double randomYcor ()

  Gets a random y coodinate of the turtles managed by this Observer.
- bool **spacePtToGridPt** (std::vector< double > &spacePt, std::vector< int > &gridPt)
- template<typename AgentType >
  AgentType * hatch (RelogoAgent *parent)

  Hatchs an agent of the specified type.
- template<typename AgentType , typename FactoryFunctor >
  AgentType * hatch (RelogoAgent *parent, FactoryFunctor agentCreator)

  Hatchs an agent of the specified type.

- template<typename AgentType >
  AgentType * who (const AgentId &id)

  Gets the agent with the specified id.
- template<typename AgentType >
  int create (size_t count)

  Create count number of agents of the specified type.
- template<typename AgentType , typename FactoryFunctor >
  int create (size_t count, FactoryFunctor agentCreator)

  Create count number of agents of the specified type, using the specified FactoryFunctor.
- void removeAgent (const AgentId &id)

  Removes the specified turtle from the world.
- template<typename AgentType >
  AgentSet< AgentType > get ()

  Gets all of the agents of the templated type and returns them in an AgentSet.
- AgentSet< Turtle > turtles ()

  Gets all the turtles in this world and return them in the AgentSet.
- void get (AgentSet< Turtle > &turtles)

  Gets all the turtles in this world and put them into the specified AgentSet.
- template<typename AgentType >
  void get (AgentSet< AgentType > &agentSet)

  Gets all of the agents of the templated type and puts them into the agentSet.
- template<typename AgentType >
  AgentType * get (const AgentId &id)

  Gets the turtle with the specified id or 0 if no such turtle is found.
- template<typename AgentType >
  AgentSet< AgentType > turtlesAt (int x, int y)

  Gets all of the agents of the templated type at the specified patch location.
- template<typename AgentType >
  void turtlesAt (int x, int y, AgentSet< AgentType > &set)

  Gets all of the agents of the templated type at the specified patch location and puts them in the specified set.
- int rank () const

  Gets the process rank of this Observer.
- const RelogoGridType * grid ()

  Gets the grid managed by this Observer.
- const RelogoSpaceType * space ()

  Gets the space managed by this Observer.
- void createLink (RelogoAgent *source, RelogoAgent *target, const std::string &networkName)

  Creates a link between the source and target agents in the named network.
- template<typename LinkCreator >
  void createLink (RelogoAgent *source, RelogoAgent *target, const std::string &networkName, LinkCreator &creator)

  Creates a link between the source and target agents in the named network, using the specified LinkCreator.
- boost::shared_ptr< RelogoLink > link (RelogoAgent *source, RelogoAgent *target, const std::string &networkName)

  Gets the link, if any, between the source and target agents in the named network.
- template<typename AgentType >
  void predecessors (RelogoAgent *agent, const std::string &networkName, AgentSet< AgentType > &out)

  Gets the network predecessors of the specified agent in the specified network and puts the result into out.
- template<typename AgentType >
  void successors (RelogoAgent *agent, const std::string &networkName, AgentSet< AgentType > &out)

  Gets the network successors of the specified agent in the specified network and puts the result into out.
- template<typename PatchType >
  PatchType * patchAt (int x, int y)

Gets the patch at the specified coordinates.

- Patch * patchAt (int x, int y)

  Gets the patch at the specified coordinates.

- Patch * patchAt (Point< double > location, double dx, double dy)

  Gets the patch at the delta from the specified location or 0 if the resulting location is outside the world.

- Patch * patchAtOffset (Point< double > location, double heading, double distance)

  Gets the patch at the heading/distance offset from the specified location or 0 if the resulting location is outside the world.

- template<typename PatchType >
  AgentSet< PatchType > patches ()

  Gets an agent set of the all the patches.

- template<typename PatchType >
  void patches (AgentSet< PatchType > &set)

  Gets all the patches and places them in the specified set.

- template<typename TurtleType >
  void turtlesOn (AgentSet< RelogoAgent > &agentSet, AgentSet< TurtleType > &out)

  Gets all the turtles that are on any patches contained in the agentSet or on the patches where any turtles in the agentSet are standing.

- template<typename TurtleType >
  void turtlesOn (const RelogoAgent *agent, AgentSet< TurtleType > &out)

  Gets all the turtles that are on the patch if the agent is a patch, otherwise get all the agents on the patch where the agent is standing.

- template<typename AgentType >
  void inRadius (const Point< double > &center, AgentSet< RelogoAgent > &inSet, double radius, AgentSet< AgentType > &outSet)

  Puts all the agents in the inSet that are of the specified type and within the specified radius from the specified center into the outSet.

- template<typename TurtleContent , typename Provider , typename Updater , typename AgentCreator >
  void synchronizeTurtleStatus (Provider &provider, Updater &updater, AgentCreator &creator, RepastProcess::EXCHANGE_PATTERN exchangePattern=RepastProcess::POLL)

  Synchronizes the status (moved or died) of all turtles across processes.

- template<typename TurtleContent , typename Provider , typename Updater >
  void synchronizeTurtleStates (Provider &provider, Updater &updater)

  Synchronizes the state of any Turtles that are shared across processes.

- template<typename TurtleContent , typename Provider , typename Updater , typename AgentCreator >
  void **synchronize** (Provider &provider, Updater &updater, AgentCreator &creator, RepastProcess::EXCHANGE_PATTERN exchangePattern=RepastProcess::POLL)

- template<typename Agent >
  AgentSet< Agent > **get** ()

Protected Types

- typedef SharedNetwork
  < RelogoAgent, RelogoLink,
  RepastEdgeContent< RelogoAgent >
  , RepastEdgeContentManager
  < RelogoAgent > > **NetworkType**

Protected Member Functions

- NetworkType * **findNetwork** (const std::string &name)

Protected Attributes

- Properties **_props**
- int **_rank**
- GridDimensions **localBounds**
- repast::SharedContext < RelogoAgent > **context**
- std::vector< repast::DataSet * > **dataSets**

Friends

- class **WorldCreator**

4.8.1 Detailed Description

Implementation of a logo Observer.

4.8.2 Member Function Documentation

4.8.2.1 void repast::relogo::Observer::_setup (Properties & *props*)

Performs internal Relogo initialization.

Parameters

props	Properties collection that can be used to drive initialization

4.8.2.2 void repast::relogo::Observer::addDataSet (repast::DataSet * *dataSet*)

Adds a dataset to this Observer.

This observer will schedule the dataset for recording and writing, and properly destroy the dataset.

Parameters

dataSet	the data set to add

4.8.2.3 template<typename AgentType > int repast::relogo::Observer::create (size_t *count*)

Create count number of agents of the specified type.

Parameters

count	the number of agents to create

Template Parameters

AgentType	the type of agents to create. This must be a Turtle or a class that extends Turtle. It must have a constructor that takes an AgentId and a pointer to this Observer.

Returns

the integer type id for agents of this type.

4.8.2.4 template<typename AgentType , typename FactoryFunctor > int repast::relogo::Observer::create (size_t *count,* FactoryFunctor *agentCreator*)

Create count number of agents of the specified type, using the specified FactoryFunctor.

Parameters

count	the number of agents to create
agentCreator	a FactoryFunctor used to create the agents

Template Parameters

AgentType	the type of agent to create. This must either be a Turtle or extend Turtle.
FactoryFunctor	a functor or function with the following signature AgentType* (AgentId id, Observer* obs).

Returns

an the integer id for agents of this type

4.8.2.5 void repast::relogo::Observer::createLink (RelogoAgent * *source*, RelogoAgent * *target*, const std::string & *networkName*)

Creates a link between the source and target agents in the named network.

Parameters

source	the source agent
target	the target agent
networkName	the name of the network to create the link in

4.8.2.6 template<typename LinkCreator > void repast::relogo::Observer::createLink (RelogoAgent * *source*, RelogoAgent * *target*, const std::string & *networkName*, LinkCreator & *creator*)

Creates a link between the source and target agents in the named network, using the specified LinkCreator.

Parameters

source	the source agent
target	the target agent
networkName	the name of the network to create the link in

Template Parameters

LinkCreator	an function or functor with the following signature RelogoLink* (Turtle* source, Turtle* target)

4.8.2.7 template<typename AgentType > AgentSet<AgentType> repast::relogo::Observer::get ( )

Gets all of the agents of the templated type and returns them in an AgentSet.

Template Parameters

AgentType	the type of turtle agents to get

Returns

an agent set containing the agents

4.8.2.8 void repast::relogo::Observer::get (AgentSet< Turtle > & *turtles*)

Gets all the turtles in this world and put them into the specified AgentSet.

Parameters

turtles	the AgentSet to put the turtles in

4.8.2.9 template<typename AgentType > void repast::relogo::Observer::get (AgentSet< AgentType > & *agentSet*)

Gets all of the agents of the templated type and puts them into the agentSet.

Parameters

agentSet	the AgentSet to put the found agents in

Template Parameters

AgentType	the type of turtle agents to get

Returns

an agent set containing the agents

4.8.2.10 template<typename AgentType > AgentType * repast::relogo::Observer::get (const AgentId & *id*)

Gets the turtle with the specified id or 0 if no such turtle is found.

Template Parameters

AgentType	the type of the turtle to find

Returns

the turtle with the specified id, or 0.

4.8.2.11 virtual void repast::relogo::Observer::go ( ) [pure virtual]

Called every tick of the simulation.

Implementations of this method will implement that actual simulation behavior.

4.8.2.12 const RelogoGridType * repast::relogo::Observer::grid ( )

Gets the grid managed by this Observer.

Returns

the grid managed by this Observer.

4.8.2.13 template<typename AgentType > AgentType * repast::relogo::Observer::hatch (RelogoAgent * *parent*)

Hatchs an agent of the specified type.

The new agent will have the location and heading of the specified "parent".

Parameters

the	"parent" of the hatched agent

Template Parameters

AgentType	the type of turtle to hatch

4.8.2.14 template<typename AgentType , typename FactoryFunctor > AgentType * repast::relogo::Observer::hatch (RelogoAgent * *parent,* FactoryFunctor *agentCreator*)

Hatchs an agent of the specified type.

The new agent will have the location and heading of the specified "parent" and will be created using the FactoryFunctor.

Parameters

the	"parent" of the hatched agent
agentCreator	a FactoryFunctor used to create the agent

Template Parameters

FactoryFunctor	a functor or function with the following signature AgentType* (AgentId id, Observer* obs).
AgentType	the type of turtle to hatch. This must either be Turtle or extend Turtle.

4.8.2.15 template<typename AgentType > void repast::relogo::Observer::inRadius (const Point< double > & *center,* AgentSet< RelogoAgent > & *inSet,* double *radius,* AgentSet< AgentType > & *outSet*)

Puts all the agents in the inSet that are of the specified type and within the specified radius from the specified center into the outSet.

Parameters

center	the center of the circle within whose radius we filter on
inSet	the set of agents to filter
outSet	the set that will contain the results of the radius filter

Template Parameters

the	type of agent to get

4.8.2.16 boost::shared_ptr< RelogoLink > repast::relogo::Observer::link (RelogoAgent * *source,* RelogoAgent * *target,* const std::string & *networkName*)

Gets the link, if any, between the source and target agents in the named network.

Parameters

source	the source of the link
target	the target of the link
networkName	the name of the network to find link in

4.8.2.17 int repast::relogo::Observer::maxPxcor ( ) const

Gets the maximum x coordinate of the patches managed by this Observer.

Returns

the maximum x coordinate of the patches managed by this Observer.

4.8.2.18 int repast::relogo::Observer::maxPycor ( ) const

Gets the maximum y coordinate of the patches managed by this Observer.

Returns

the maximum y coordinate of the patches managed by this Observer.

4.8.2.19 int repast::relogo::Observer::minPxcor ( ) const

Gets the minimum x coordinate of the patches managed by this Observer.

Returns

the minimum x coordinate of the patches managed by this Observer.

4.8.2.20 int repast::relogo::Observer::minPycor ( ) const

Gets the minimum y coordinate of the patches managed by this Observer.

Returns

the minimum y coordinate of the patches managed by this Observer.

4.8.2.21 template<typename PatchType > PatchType * repast::relogo::Observer::patchAt (int *x*, int *y*)

Gets the patch at the specified coordinates.

Parameters

x	the x coordinate
y	the y coordinate

Template Parameters

the	patch type

Returns

a pointer to the patch at x,y

4.8.2.22 Patch* repast::relogo::Observer::patchAt (int *x*, int *y*)

Gets the patch at the specified coordinates.

Parameters

x	the x coordinate
y	the y coordinate

Returns

a pointer to the patch at x,y

4.8.2.23 Patch * repast::relogo::Observer::patchAt (Point< double > *location,* double *dx,* double *dy*)

Gets the patch at the delta from the specified location or 0 if the resulting location is outside the world.

Parameters

location	
dx	the delta along the x dimension
dy	the delta along the y dimension

Returns

the patch at the delta from the specified location or 0 if the resulting location is outside the world.

4.8.2.24 Patch * repast::relogo::Observer::patchAtOffset (Point< double > *location*, double *heading*, double *distance*)

Gets the patch at the heading/distance offset from the specified location or 0 if the resulting location is outside the world.

Parameters

location	
heading	the heading away from location
distance	distance along heading

Returns

the patch at the delta from the specified location or 0 if the resulting location is outside the world.

4.8.2.25 template<typename PatchType > AgentSet< PatchType > repast::relogo::Observer::patches ( )

Gets an agent set of the all the patches.

Template Parameters

PatchType	the patch type

Returns

all the patches.

4.8.2.26 template<typename PatchType > void repast::relogo::Observer::patches (AgentSet< PatchType > & *set*)

Gets all the patches and places them in the specified set.

Parameters

set	the AgentSet to put the patches in

Template Parameters

PatchType	the patch type

4.8.2.27 template<typename AgentType > void repast::relogo::Observer::predecessors (RelogoAgent * *agent*, const std::string & *networkName*, AgentSet< AgentType > & *out*)

Gets the network predecessors of the specified agent in the specified network and puts the result into out.

Parameters

agent	the agent to get the predecessors of
networkName	the name of the network
out	an AgentSet when the predecessors will be put

Template Parameters

AgentType	the type of the predecessors

4.8.2.28 int repast::relogo::Observer::randomPxcor ( )

Gets a random x coodinate of the patches managed by this Observer.

Returns

a random x coordinate of the patches managed by this Observer.

4.8.2.29 int repast::relogo::Observer::randomPycor ( )

Gets a random y coodinate of the patches managed by this Observer.

Returns

a random y coordinate of the patches managed by this Observer.

4.8.2.30 double repast::relogo::Observer::randomXcor ( )

Gets a random x coodinate of the turtles managed by this Observer.

Returns

a random x coordinate of the turtles managed by this Observer.

4.8.2.31 double repast::relogo::Observer::randomYcor ( )

Gets a random y coodinate of the turtles managed by this Observer.

Returns

a random y coordinate of the turtles managed by this Observer.

4.8.2.32 int repast::relogo::Observer::rank ( ) const [inline]

Gets the process rank of this Observer.

Returns

the process rank of this Observer.

4.8.2.33 virtual void repast::relogo::Observer::setup (Properties & *props*) [inline], [virtual]

Classes that extend this should include model initialization here.

Parameters

props	Properties collection that can be used to drive initialization

4.8.2.34 const RelogoSpaceType * repast::relogo::Observer::space ( )

Gets the space managed by this Observer.

Returns

the space managed by this Observer.

4.8.2.35 template<typename AgentType > void repast::relogo::Observer::successors (RelogoAgent * *agent*, const std::string & *networkName*, AgentSet< AgentType > & *out*)

Gets the network successors of the specified agent in the specified network and puts the result into out.

Parameters

agent	the agent to get the successors of
networkName	the name of the network
out	an AgentSet when the successors will be put

Template Parameters

AgentType	the type of the successors

4.8.2.36 template<typename TurtleContent , typename Provider , typename Updater > void repast::relogo::Observer::synchronizeTurtleStates (Provider & *provider*, Updater & *updater*)

Synchronizes the state of any Turtles that are shared across processes.

If no turtles are shared across processes, then this does not need to be called.

Parameters

provider	provides TurtleContent given an AgentRequest
updater	updates an existing agent given TurtleContent

Template Parameters

TurtleContent	the serializable struct or class that describes the state of turtles and patches
Provider	given an AgentRequest, a Provider provides the TurtleContent for the requested Turtles, implementing void provideContent(const AgentRequest&, std::vector<-TurtleContent>&)
Updater	given TurtleContent, an Updater updates an existing agent with the TurtleContent, implementing void updateAgent(const TurtleContent&).

4.8.2.37 template<typename TurtleContent , typename Provider , typename Updater , typename AgentCreator > void repast::relogo::Observer::synchronizeTurtleStatus (Provider & *provider*, Updater & *updater*, AgentCreator & *creator*, RepastProcess::EXCHANGE_PATTERN *exchangePattern* = `RepastProcess::POLL`)

Synchronizes the status (moved or died) of all turtles across processes.

If any turtle may have moved into the grid portion managed by another process or if any turtle has died then this must be called prior to those turtles doing anything.

Parameters

provider	the class that provides agents given an AgentRequest
creator	creates Turtles given TurtleContent

Template Parameters

TurtleContent	the serializable struct or class that describes a turtles state.
Provider	a class that provides TurtleContent from given an AgentRequest, implementing void provideContent(const repast::AgentRequest&, std::vector<TurtleContent>& out)
AgentCreator	a class that can create agents from TurtleContent, implementing RelogoAgent* createAgent(TurtleContent&).

4.8.2.38 template<typename AgentType > AgentSet< AgentType > repast::relogo::Observer::turtlesAt (int *x*, int *y*)

Gets all of the agents of the templated type at the specified patch location.

Template Parameters

AgentType	the type of the agents

Returns

an agent list containing the agents at the specified location.

4.8.2.39 template<typename AgentType > void repast::relogo::Observer::turtlesAt (int *x*, int *y*, AgentSet< AgentType > & *set*)

Gets all of the agents of the templated type at the specified patch location and puts them in the specified set.

Parameters

x	the x coordinate of the patch
y	the y coordinate of the patch
set	the AgentSet to add the found agents to

Template Parameters

AgentType	the agent type

4.8.2.40 template<typename TurtleType > void repast::relogo::Observer::turtlesOn (AgentSet< RelogoAgent > & *agentSet*, AgentSet< TurtleType > & *out*)

Gets all the turtles that are on any patches contained in the agentSet or on the patches where any turtles in the agentSet are standing.

The result is placed in out.

Parameters

agentSet	a set of turtles or patches
out	the AgentSet where the found turtles will put

Template Parameters

TurtleType	the type of the turtles to return

4.8.2.41 template<typename TurtleType > void repast::relogo::Observer::turtlesOn (const RelogoAgent * *agent*, AgentSet< TurtleType > & *out*)

Gets all the turtles that are on the patch if the agent is a patch, otherwise get all the agents on the patch where the agent is standing.

The result is placed in out.

Parameters

agent	the turtle or patch used to determine which turtles to get
out	the agent set where the found turtles will be put

Template Parameters

TurtleType	the type of the turtles to return

4.8.2.42 template<typename AgentType > AgentType * repast::relogo::Observer::who (const AgentId & *id*)

Gets the agent with the specified id.

Parameters

id	the id of the agent to get

Template Parameters

AgentType	the type of the agent to get

Returns

the agent with the specified id, or 0 if the agent is not found.

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

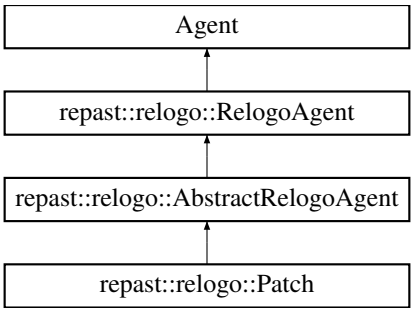
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Observer.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Observer.cpp

4.9 repast::relogo::Patch Class Reference

A logo patch.

`#include <Patch.h>`

Inheritance diagram for repast::relogo::Patch:



Public Member Functions

- Patch (repast::AgentId id, Observer *observer)

  Creates a Patch that will have the specified id and be managed by the specified Observer.
- virtual int pxCor () const

  Gets the patch x coordinate of this patch's location.
- virtual int pyCor () const

  Gets the patch y coordinate of this patch's location.
- template<typename PatchType >
  AgentSet< PatchType > neighbors ()

  Gets the 8 (Moore neighborhood) neighboring Patches of this Patch.
- template<typename PatchType >
  AgentSet< PatchType > neighbors4 ()

  Gets the 4 (Von Neumann neighborhood) neighboring Patches of this Patch.
- template<typename PatchType >
  void neighbors (AgentSet< PatchType > &out)

  Gets the 8 (Moore neighborhood) neighboring Patches of this Patch and puts them out.
- template<typename PatchType >
  void neighbors4 (AgentSet< PatchType > &out)

  Gets the 4 (Von Neumann neighborhood) neighboring Patches of this Patch.

4.9.1 Detailed Description

A logo patch.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 repast::relogo::Patch::Patch (repast::AgentId *id,* Observer * *observer*)

Creates a Patch that will have the specified id and be managed by the specified Observer.

Parameters

id	
observer	

4.9.3 Member Function Documentation

4.9.3.1 template<typename PatchType > AgentSet< PatchType > repast::relogo::Patch::neighbors ( )

Gets the 8 (Moore neighborhood) neighboring Patches of this Patch.

Template Parameters

the	patch type

Returns

the 8 (Moore neighborhood) neighboring Patches of this Patch.

4.9.3.2 template<typename PatchType > void repast::relogo::Patch::neighbors (AgentSet< PatchType > & *out*)

Gets the 8 (Moore neighborhood) neighboring Patches of this Patch and puts them out.

Parameters

out	the AgentSet to the neighbors in

Template Parameters

the	patch type

4.9.3.3 template<typename PatchType > AgentSet< PatchType > repast::relogo::Patch::neighbors4 ( )

Gets the 4 (Von Neumann neighborhood) neighboring Patches of this Patch.

Template Parameters

the	patch type

Returns

the 4 (Von Neumann neighborhood) neighboring Patches of this Patch.

4.9.3.4 template<typename PatchType > void repast::relogo::Patch::neighbors4 (AgentSet< PatchType > & *out*)

Gets the 4 (Von Neumann neighborhood) neighboring Patches of this Patch.

Parameters

out	the AgentSet to put the neighbors in

Template Parameters

the	patch type

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Patch.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Patch.cpp

4.10 repast::relogo::RandomMove Class Reference

Operator(() that implements random move for a turtle.

`#include <RandomMove.h>`

Public Member Functions

- RandomMove (Observer *observer)

  Creates a RandomMove that randomly move turtles within the spaces managed by the specified observer.
- void operator() (Turtle *turtle) const

  Move the specified turtle at random.

4.10.1 Detailed Description

Operator(() that implements random move for a turtle.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 repast::relogo::RandomMove::RandomMove (Observer * *observer*) [inline]

Creates a RandomMove that randomly move turtles within the spaces managed by the specified observer.

Parameters

observer	the observer whose spaces and turtles we want to move

4.10.3 Member Function Documentation

4.10.3.1 void repast::relogo::RandomMove::operator() (Turtle * *turtle*) const

Move the specified turtle at random.

Parameters

turtle	the turtel to move

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

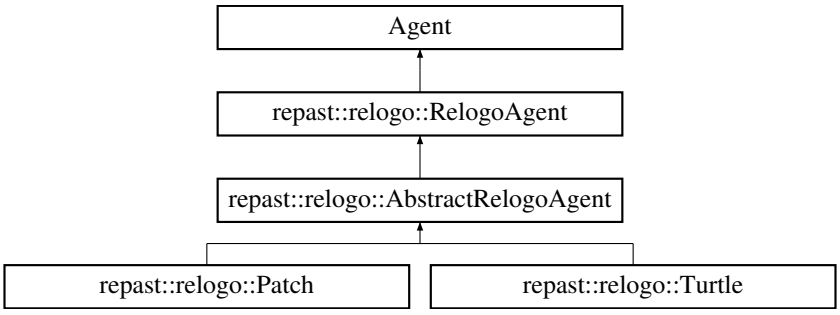
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RandomMove.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RandomMove.cpp

4.11 repast::relogo::RelogoAgent Class Reference

Base agent for Relogo.

`#include <RelogoAgent.h>`

Inheritance diagram for repast::relogo::RelogoAgent:



Public Member Functions

- RelogoAgent (repast::AgentId id, Observer *observer)

  Creates a RelogoAgent with the specified id and managed by the specified Observer.
- virtual repast::AgentId & getId ()

  Gets the id of this RelogoAgent.
- virtual const repast::AgentId & getId () const

  Gets the const id of this RelogoAgent.
- Point< double > location () const

  Gets the location of this RelogoAgent.
- virtual void hatchCopy ()

  If this ReLogo agent is 'hatched', makes an appropriate copy, setting instance variables as appropriate.
- double xCor () const

  Gets the x coordinate of the agent's location.
- double yCor () const

  Gets the y coordinate of the agent's location.
- virtual int pxCor () const =0

  Gets the patch x coordinate of the agent's location.
- virtual int pyCor () const =0

  Gets the patch y coordinate of the agent's location.
- double distance (RelogoAgent *obj) const

  Gets the distance from this RelogoAgent to the specified agent.
- double distancexy (double x, double y) const

  Gets the distance from this RelogoAgent to the specified point.

Protected Attributes

- Observer * **_observer**
- Point< double > **_location**
- repast::AgentId **_id**

Friends

- class **RelogoContinuousSpaceAdder**
- class **WorldCreator**
- template<typename GPTransformer , typename Adder >
  class **RelogoSharedContinuousSpace**

4.11.1 Detailed Description

Base agent for Relogo.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 repast::relogo::RelogoAgent::RelogoAgent (repast::AgentId *id,* Observer * *observer*) [inline]

Creates a RelogoAgent with the specified id and managed by the specified Observer.

Parameters

id	the id of this RelogoAgent
observer	the observer that will manage this agent.

4.11.3 Member Function Documentation

4.11.3.1 double repast::relogo::RelogoAgent::distance (RelogoAgent * *obj*) const

Gets the distance from this RelogoAgent to the specified agent.

Returns

the distance from this RelogoAgent to the specified agent.

4.11.3.2 double repast::relogo::RelogoAgent::distancexy (double *x*, double *y*) const

Gets the distance from this RelogoAgent to the specified point.

Returns

the distance from this RelogoAgent to the specified point.

4.11.3.3 virtual repast::AgentId& repast::relogo::RelogoAgent::getId ( ) [inline], [virtual]

Gets the id of this RelogoAgent.

Returns

the id of this RelogoAgent.

4.11.3.4 virtual const repast::AgentId& repast::relogo::RelogoAgent::getId ( ) const [inline], [virtual]

Gets the const id of this RelogoAgent.

Returns

the const id of this RelogoAgent.

4.11.3.5 Point<double> repast::relogo::RelogoAgent::location ( ) const [inline]

Gets the location of this RelogoAgent.

Returns

the location of this RelogoAgent.

4.11.3.6 virtual int repast::relogo::RelogoAgent::pxCor ( ) const [pure virtual]

Gets the patch x coordinate of the agent's location.

Returns

the patch x coordinate of the agent's location.

Implemented in repast::relogo::Turtle, repast::relogo::Patch, and repast::relogo::AbstractRelogoAgent.

4.11.3.7 virtual int repast::relogo::RelogoAgent::pyCor ( ) const [pure virtual]

Gets the patch y coordinate of the agent's location.

Returns

the patch y coordinate of the agent's location.

Implemented in repast::relogo::Turtle, repast::relogo::Patch, and repast::relogo::AbstractRelogoAgent.

4.11.3.8 double repast::relogo::RelogoAgent::xCor ( ) const

Gets the x coordinate of the agent's location.

Returns

the x coordinate of the agent's location.

4.11.3.9 double repast::relogo::RelogoAgent::yCor ( ) const

Gets the y coordinate of the agent's location.

Returns

the y coordinate of the agent's location.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoAgent.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoAgent.cpp

4.12 repast::relogo::RelogoContinuousSpaceAdder Class Reference

An "Adder" for adding RelogoAgents to RelogoSpaces.

`#include <RelogoContinuousSpaceAdder.h>`

Public Member Functions

- void **init** (GridDimensions dimensions, RelogoSpaceType *grid)
- bool **add** (boost::shared_ptr< RelogoAgent > agent)

4.12.1 Detailed Description

An "Adder" for adding RelogoAgents to RelogoSpaces.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoContinuousSpaceAdder.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoContinuousSpaceAdder.cpp

4.13 repast::relogo::RelogoDiscreteSpaceAdder Class Reference

Adds RelogoAgents to RelogoDiscreteSpaces.

`#include <RelogoDiscreteSpaceAdder.h>`

Public Member Functions

- void **init** (GridDimensions dimensions, RelogoGridType *grid)
- bool **add** (boost::shared_ptr< RelogoAgent > agent)

4.13.1 Detailed Description

Adds RelogoAgents to RelogoDiscreteSpaces.

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

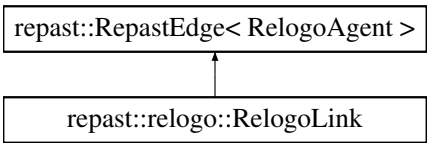
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoDiscreteSpaceAdder.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoDiscreteSpaceAdder.cpp

4.14 repast::relogo::RelogoLink Class Reference

Network link for Relogo.

`#include <RelogoLink.h>`

Inheritance diagram for repast::relogo::RelogoLink:



Public Member Functions

- RelogoLink ()

  Creates an empty RelogoLink with no source or target.
- RelogoLink (RelogoAgent *source, RelogoAgent *target)

  Creates a RelogoLink with the specified source and target and a default weight of 1.
- RelogoLink (RelogoAgent *source, RelogoAgent *target, double weight)

  Creates a RelogoLink with the specified source, target, and weight.
- RelogoLink (boost::shared_ptr< RelogoAgent > source, boost::shared_ptr< RelogoAgent > target)

  Creates a RelogoLink with the specified source and target and a default weight of 1.
- RelogoLink (boost::shared_ptr< RelogoAgent > source, boost::shared_ptr< RelogoAgent > target, double weight)

  Creates a RelogoLink with the specified source, target, and weight.
- RelogoLink (const RelogoLink &edge)

  Copy constructor that creates a RelogoLink from another RelogoLink.

4.14.1 Detailed Description

Network link for Relogo.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 repast::relogo::RelogoLink::RelogoLink (RelogoAgent * *source*, RelogoAgent * *target*)

Creates a RelogoLink with the specified source and target and a default weight of 1.

Parameters

source	the link source
target	the link target

4.14.2.2 repast::relogo::RelogoLink::RelogoLink (RelogoAgent * *source*, RelogoAgent * *target*, double *weight*)

Creates a RelogoLink with the specified source, target, and weight.

Parameters

source	the link source
target	the link target
weight	the link weight

4.14.2.3 repast::relogo::RelogoLink::RelogoLink (boost::shared_ptr< RelogoAgent > *source*, boost::shared_ptr< RelogoAgent > *target*)

Creates a RelogoLink with the specified source and target and a default weight of 1.

Parameters

source	the link source
target	the link target

4.14.2.4 repast::relogo::RelogoLink::RelogoLink (boost::shared_ptr< RelogoAgent > *source*, boost::shared_ptr< RelogoAgent > *target*, double *weight*)

Creates a RelogoLink with the specified source, target, and weight.

Parameters

source	the link source
target	the link target
weight	the link weight

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

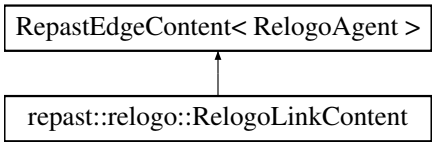
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoLink.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoLink.cpp

4.15 repast::relogo::RelogoLinkContent Class Reference

Subclass of RepastEdgeContent, used in synchronization.

`#include <RelogoLink.h>`

Inheritance diagram for repast::relogo::RelogoLinkContent:



Public Member Functions

- **RelogoLinkContent** (RelogoLink *link)

4.15.1 Detailed Description

Subclass of RepastEdgeContent, used in synchronization.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoLink.h

4.16 repast::relogo::RelogoLinkContentManager Class Reference

Subclass of RepastEdgeContentManager, used to package and rebuild edges during synchronization.

`#include <RelogoLink.h>`

Public Member Functions

- RelogoLink * **createEdge** (RelogoLinkContent &content, repast::Context< RelogoAgent > *context)
- RelogoLinkContent * **provideEdgeContent** (RelogoLink *edge)

4.16.1 Detailed Description

Subclass of RepastEdgeContentManager, used to package and rebuild edges during synchronization.

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

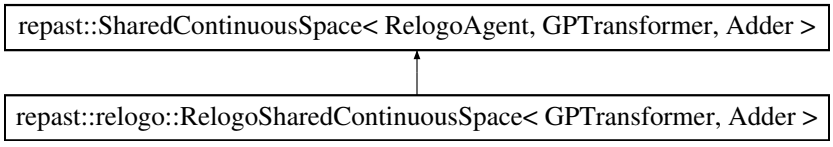
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoLink.h

4.17 repast::relogo::RelogoSharedContinuousSpace< GPTransformer, Adder > Class Template Reference

Repast SharedContinuousSpace specialized for Relogo.

`#include <RelogoSharedContinuousSpace.h>`

Inheritance diagram for repast::relogo::RelogoSharedContinuousSpace< GPTransformer, Adder >:



Public Member Functions

- **RelogoSharedContinuousSpace** (std::string name, repast::GridDimensions gridDims, std::vector< int > processDims, int buffer, boost::mpi::communicator *world)

Protected Member Functions

- void **synchMoveTo** (const repast::AgentId &id, const repast::Point< double > &pt)

4.17.1 Detailed Description

template<typename GPTransformer, typename Adder>class repast::relogo::RelogoSharedContinuousSpace< GPTransformer, Adder >

Repast SharedContinuousSpace specialized for Relogo.

This overrides synchMoveTo.

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

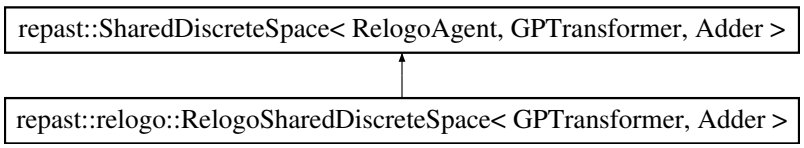
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoAgent.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoSharedContinuousSpace.h

4.18 repast::relogo::RelogoSharedDiscreteSpace< GPTransformer, Adder > Class Template Reference

Repast SharedDiscreteSpace specialized for Relogo.

`#include <RelogoSharedDiscreteSpace.h>`

Inheritance diagram for repast::relogo::RelogoSharedDiscreteSpace< GPTransformer, Adder >:



Public Member Functions

- **RelogoSharedDiscreteSpace** (std::string name, repast::GridDimensions gridDims, std::vector< int > processDims, int buffer, boost::mpi::communicator *world)

4.18.1 Detailed Description

template<typename GPTransformer, typename Adder>class repast::relogo::RelogoSharedDiscreteSpace< GPTransformer, Adder >

Repast SharedDiscreteSpace specialized for Relogo.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/RelogoSharedDiscreteSpace.h

4.19 repast::relogo::SetCmp< T, ValueGetter > Struct Template Reference

Compares two items using the specified getter.

`#include <AgentSet.h>`

Public Member Functions

- **SetCmp** (const ValueGetter *getter)
- bool **operator()** (T *one, T *two)

Public Attributes

- const ValueGetter * **_getter**

4.19.1 Detailed Description

template<typename T, typename ValueGetter>struct repast::relogo::SetCmp< T, ValueGetter >

Compares two items using the specified getter.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/AgentSet.h

4.20 repast::relogo::SimulationRunner Class Reference

Runs a Relogo simulation.

`#include <SimulationRunner.h>`

Public Member Functions

- SimulationRunner (boost::mpi::communicator *world)

  Creates a SimulationRunner.

- template<typename ObserverType , typename PatchType >
  void run (Properties &props)

  Creates and runs the simulation using the properties defined in props.

Protected Attributes

- boost::mpi::communicator * **comm**

4.20.1 Detailed Description

Runs a Relogo simulation.

4.20.2 Member Function Documentation

4.20.2.1 template<typename ObserverType , typename PatchType > void repast::relogo::SimulationRunner::run (Properties & *props*)

Creates and runs the simulation using the properties defined in props.

The properties file must have the following properties defined:

- min.x the minimum integer x coordinate of the world
- min.y the minimum integer y coordinate of the world
- max.x the maximum integer x coordinate of the world
- max.h the maximum integer y coordinate of the world
- grid.buffer the size of the grid and space buffers
- proc.per.x the number of processes to assign to the world's x dimension. proc.per.x multiplied by proc.per.y must equal the number processes that the simulation will run on
- proc.per.y the number of processes to assign to the world's y dimension. proc.per.x multiplied by proc.per.y must equal the number processes that the simulation will run on
- stop.at the tick at which to stop the simulation

  This will create an Observer of the specified type and populate the world with Patches of the specified type. It will then call setup(props) on that Observer implementation and start the simulation schedule which will call the Observer's go method each tick.

Parameters

props	a properties file containing the properties mentioned above

Template Parameters

ObserverType	the type of Observer to create. This type must extend relogo::Observer.
PatchType	the type of Patches to create. This must extend relogo::Patch.

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

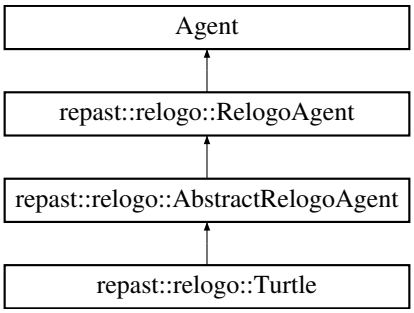
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/SimulationRunner.h

4.21 repast::relogo::Turtle Class Reference

Relogo Turtle implementation.

`#include <Turtle.h>`

Inheritance diagram for repast::relogo::Turtle:



Public Member Functions

- Turtle (repast::AgentId id, Observer *observer)

  Creates a Turtle that will have the specified id, and be managed by the specified Observer.
- virtual void **hatchCopy** (RelogoAgent *parent)
- void xCor (double x)

  Sets the x coordinate of the Turtle's location.
- void yCor (double y)

  Sets the y coordinate of the Turtle's location.
- void setxy (double x, double y)

  Sets the x and y coordinate of the Turtle's location.
- virtual int pxCor () const

  Gets the Patch x coordinate of this Turtle.
- virtual int pyCor () const

  Gets the Patch x coordinate of this Turtle.
- void die ()

  Removes this turtle from the world.
- void createLinkWith (Turtle *turtle, const std::string &network=DEFAULT_UNDIR_NET)

  Creates a link between this turtle and the specified turtle in the specified undirected network.
- template<typename LinkCreator >
  void createLinkWithLC (Turtle *turtle, LinkCreator &creator, const std::string &network=DEFAULT_UNDIR_NET)

  Creates a link between this turtle and the specified turtle in the specified undirected network, using the specified linke creator.
- template<typename AgentType >
  void createLinksWith (AgentSet< AgentType > &agents, const std::string &network=DEFAULT_UNDIR_NET)

  Creates links between this turtle and all the agents in the AgentSet in the named network.
- template<typename AgentType , typename LinkCreator >
  void createLinksWithLC (AgentSet< AgentType > &agents, LinkCreator &creator, const std::string &network=DEFAULT_UNDIR_NET)

  Creates links between this turtle and all the agents in the agentset using the link creator and in the named network.
- void createLinkFrom (Turtle *turtle, const std::string &network=DEFAULT_DIR_NET)

  Creates a link to this Turtle from the specified turtle in the named network which defaults to the default directed network.
- template<typename LinkCreator >
  void createLinkFromLC (Turtle *turtle, LinkCreator &linkCreator, const std::string &network=DEFAULT_DIR_NET)

  Creates a link to this Turtle from the specified turtle in the named network which defaults to the default directed network.
- template<typename AgentType >
  void createLinksFrom (AgentSet< AgentType > &agents, const std::string &network=DEFAULT_DIR_NET)

  Creates links to this turtle from all the agents in the agentset in the named network.
- template<typename AgentType , typename LinkCreator >
  void createLinksFromLC (AgentSet< AgentType > &agents, LinkCreator &creator, const std::string &network=DEFAULT_DIR_NET)

  Creates links to this turtle from all the agents in the agentset using the link creator and in the named network.
- template<typename AgentType >
  void createLinksTo (AgentSet< AgentType > &agents, const std::string &network=DEFAULT_DIR_NET)

  Creates links from this turtle to all the agents in the agentset in the named network.
- template<typename AgentType , typename LinkCreator >
  void createLinksToLC (AgentSet< AgentType > &agents, LinkCreator &creator, const std::string &network=DEFAULT_DIR_NET)

Creates links from this turtle to all the agents in the agentset using the link creator and in the named network.

- void createLinkTo (Turtle *turtle, const std::string &network=DEFAULT_DIR_NET)

  Creates a link from this Turtle to the specified turtle in the named network which defaults to the default directed network.

- template<typename LinkCreator >
  void createLinkToLC (Turtle *turtle, LinkCreator &linkCreator, const std::string &network=DEFAULT_DIR_NET)

  Creates a link from this Turtle to the specified turtle in the named network which defaults to the default directed network.

- boost::shared_ptr< RelogoLink > inLinkFrom (Turtle *turtle, const std::string &name=DEFAULT_DIR_NET)

  Gets the link from the specified turtle to this one in the specified network which defaults to the default directed network.

- boost::shared_ptr< RelogoLink > outLinkTo (Turtle *turtle, const std::string &name=DEFAULT_DIR_NET)

  Gets the link from the this turtle to the specified turtle in the specified network which defaults to the default directed network.

- boost::shared_ptr< RelogoLink > linkWith (Turtle *turtle, const std::string &name=DEFAULT_UNDIR_NET)

  Gets the link between this turtle and the specified on in the named undirected network.

- bool linkNeighborQ (Turtle *turtle, const std::string &name=DEFAULT_UNDIR_NET)

  Gets whether or not this turtle is linked to the specified turtle, in the specified network.

- template<typename AgentType >
  void linkNeighbors (AgentSet< AgentType > &out, const std::string &name=DEFAULT_UNDIR_NET)

  Gets all the network neighbors of this turtle in the named network and puts them in the specified AgentSet.

- bool inLinkNeighborQ (Turtle *turtle, const std::string &name=DEFAULT_DIR_NET)

  Gets whether or not there is an edge into this turtle from the specified turtle, in the specified network.

- template<typename AgentType >
  void inLinkNeighbors (AgentSet< AgentType > &out, const std::string &name=DEFAULT_DIR_NET)

  Gets all the network predecessors of this turtle in the named network and puts them in the specified array list.

- bool outLinkNeighborQ (Turtle *turtle, const std::string &name=DEFAULT_DIR_NET)

  Gets whether or not there is an edge from this turtle to the specified turtle, in the specified network.

- template<typename AgentType >
  void outLinkNeighbors (AgentSet< AgentType > &out, const std::string &name=DEFAULT_DIR_NET)

  Gets all the network successors of this turtle in the named network and puts them in the specified array list.

- void moveTo (Turtle *turtle)

  Moves this turtle to the location of the specified turtle.

- void moveTo (Patch *patch)

  Moves this turtle to the location of the specified patch.

- void move (double distance)

  Moves this turtle the specified distance along the current heading.

- void mv (double distance)

  Moves this turtle the specified distance along the current heading.

- void jump (double distance)

  Moves this turtle forward the specified distance, if and only if that would not take this turtle outside the current topology.

- void forward (double distance)

  Moves this turtle forward the specified distance.

- void backward (double distance)

  Moves this turtle backward the specified distance.

- void fd (double distance)

  Moves this turtle forward the specified distance.

- void bk (double distance)

  Moves this turtle backward the specified distance.

- template<typename PatchType , typename ValueGetter >
  void downhill (ValueGetter &getter)

  Moves this turtle to a neighboring patch with lowest value as retrieved via the ValueGetter.

- template<typename PatchType , typename ValueGetter >
  void downhill4 (ValueGetter &getter)

  Moves this turtle to the patch with lowest value as retrieved via the ValueGetter.
- template<typename PatchType , typename ValueGetter >
  void uphill (ValueGetter &getter)

  Moves this turtle to the patch with highest value as retrieved via the ValueGetter.
- template<typename PatchType , typename ValueGetter >
  void uphill4 (ValueGetter &getter)

  Moves this turtle to the patch with highest value as retrieved via the ValueGetter.
- double dx () const

  Gets the distance traveled along the x dimension if the turtle were to take one step forward along its current heading.
- double dy () const

  Gets the distance traveled along the y dimension if the turtle were to take one step forward along its current heading.
- bool canMoveQ (double distance) const

  Gets whether or not this turtle can move the specified distance along its current heading given the current topology.
- float towards (RelogoAgent *agent) const

  Gets the heading from this turtle to the specified RelogoAgent (turtle or patch).
- float towardsxy (double x, double y) const

  Gets the heading from this turtle to the specified location.
- float towards (const Point< double > &location) const

  Gets the heading from this turtle to the specified location.
- double distance (Turtle *turtle) const

  Gets the distance from this turtle to the specified turtle.
- float heading () const

  Gets this Turtle's current heading.
- void heading (float heading)

  Sets this turtle's heading to the specified heading.
- template<typename PatchType >
  PatchType * patchHere () const

  Gets the patch under this turtle.
- void face (Turtle *turtle)

  Sets the turtles heading to face towards the specified turtle.
- void face (Patch *patch)

  Sets the turtles heading to face towards the specified pach.
- void facexy (double nx, double ny)

  Sets the turtles heading to face the specified coordinates.
- void left (float degrees)

  Turns the turtle left by the specified number of degrees.
- void lt (float degrees)

  Turns the turtle left by the specified number of degrees.
- template<typename PatchType >
  PatchType * patchLeftAndAhead (float angleInDegrees, double distance)

  Gets the patch that is the specified distance from this turtle, at the specified angle (turning left) from this turtle's heading.
- template<typename PatchType >
  PatchType * patchRightAndAhead (float angleInDegrees, double distance)

  Gets the patch that is the specified distance from this turtle, in the specified degrees (turning right) from this turtle's heading.

Additional Inherited Members

4.21.1 Detailed Description

Relogo Turtle implementation.

4.21.2 Member Function Documentation

4.21.2.1 void repast::relogo::Turtle::backward (double *distance*) [inline]

Moves this turtle backward the specified distance.

Parameters

distance	the distance to move

4.21.2.2 void repast::relogo::Turtle::bk (double *distance*) [inline]

Moves this turtle backward the specified distance.

Parameters

distance	the distance to move

4.21.2.3 bool repast::relogo::Turtle::canMoveQ (double *distance*) const

Gets whether or not this turtle can move the specified distance along its current heading given the current topology.

Returns

true if this turtle can move the specified distance along its current heading given the current topology, otherwise false

4.21.2.4 void repast::relogo::Turtle::createLinkFrom (Turtle * *turtle,* const std::string & *network* = DEFAULT_DIR_NET)

Creates a link to this Turtle from the specified turtle in the named network which defaults to the default directed network.

Parameters

turtle	the turtle that will be the source turtle of the lin
network	the name of the network

4.21.2.5 template<typename LinkCreator > void repast::relogo::Turtle::createLinkFromLC (Turtle * *turtle,* LinkCreator & *linkCreator,* const std::string & *network* = DEFAULT_DIR_NET)

Creates a link to this Turtle from the specified turtle in the named network which defaults to the default directed network.

Parameters

turtle	the turtle that will be the source turtle of the link
network	the name of the network
linkCreator	an object used to create the link

Template Parameters

LinkCreator	an function or functor with the following signature RelogoLink* (Turtle* source, Turtle* target)

4.21.2.6 template<typename AgentType > void repast::relogo::Turtle::createLinksFrom (AgentSet< AgentType > & *agents,* const std::string & *network* = DEFAULT_DIR_NET)

Creates links to this turtle from all the agents in the agentset in the named network.

The network defaults to the default directed network.

Parameters

agents	the agentset of agents to create links from
network	the name of the network to create the links in. This defaults to the default directed network

Template Parameters

AgentType	the type of object contained by the agentset.

4.21.2.7 template<typename AgentType , typename LinkCreator > void repast::relogo::Turtle::createLinksFromLC (AgentSet< AgentType > & *agents,* LinkCreator & *creator,* const std::string & *network* = DEFAULT_DIR_NET)

Creates links to this turtle from all the agents in the agentset using the link creator and in the named network.

The network defaults to the default directed network.

Parameters

agents	the agentset of agents to create links from
network	the name of the network to create the links in. This defaults to the default directed network

Template Parameters

AgentType	the type of object contained by the agentset
LinkCreator	an function or functor with the following signature RelogoLink* (Turtle* source, Turtle* target)

4.21.2.8 template<typename AgentType > void repast::relogo::Turtle::createLinksTo (AgentSet< AgentType > & *agents,* const std::string & *network* = DEFAULT_DIR_NET)

Creates links from this turtle to all the agents in the agentset in the named network.

The network defaults to the default directed network.

Parameters

agents	the agentset of agents to create links to
network	the name of the network to create the links in. This defaults to the default directed network

Template Parameters

AgentType	the type of object contained by the agentset.

4.21.2.9 template<typename AgentType , typename LinkCreator > void repast::relogo::Turtle::createLinksToLC (AgentSet< AgentType > & *agents,* LinkCreator & *creator,* const std::string & *network* = DEFAULT_DIR_NET)

Creates links from this turtle to all the agents in the agentset using the link creator and in the named network.

The network defaults to the default directed network.

Parameters

agents	the agentset of agents to create links to
network	the name of the network to create the links in. This defaults to the default directed network

Template Parameters

AgentType	the type of object contained by the agentset
LinkCreator	an function or functor with the following signature boost::shared_ptr<RelogoLink> (Turtle* source, Turtle* target)

4.21.2.10 template<typename AgentType > void repast::relogo::Turtle::createLinksWith (AgentSet< AgentType > & *agents*, const std::string & *network* = DEFAULT_UNDIR_NET)

Creates links between this turtle and all the agents in the AgentSet in the named network.

The network defaults to the default undirected network.

Parameters

agents	the agentset of agents to create links with
network	the name of the network to create the links in. This defaults to the default undirected network

Template Parameters

Agent	the type of object contained by the agentset.

4.21.2.11 template<typename AgentType , typename LinkCreator > void repast::relogo::Turtle::createLinksWithLC (AgentSet< AgentType > & *agents*, LinkCreator & *creator*, const std::string & *network* = DEFAULT_UNDIR_NET)

Creates links between this turtle and all the agents in the agentset using the link creator and in the named network.

The network defaults to the default undirected network.

Parameters

agents	the agentset of agents to create links with
network	the name of the network to create the links in. This defaults to the default undirected network
creator	the functor to create the links with

Template Parameters

Agent	the type of object contained by the agentset
LinkCreator	the object used to create the links
LinkCreator	an function or functor with the following signature RelogoLink* (Turtle* source, Turtle* target)

4.21.2.12 void repast::relogo::Turtle::createLinkTo (Turtle * *turtle*, const std::string & *network* = DEFAULT_DIR_NET)

Creates a link from this Turtle to the specified turtle in the named network which defaults to the default directed network.

Parameters

turtle	the turtle that will be the target turtle of the link
network	the name of the network

4.21.2.13 template<typename LinkCreator > void repast::relogo::Turtle::createLinkToLC (Turtle * *turtle*, LinkCreator & *linkCreator*, const std::string & *network* = DEFAULT_DIR_NET)

Creates a link from this Turtle to the specified turtle in the named network which defaults to the default directed network.

Parameters

turtle	the turtle that will be the target turtle of the link
network	the name of the network
linkCreator	an object used to create the link

Template Parameters

LinkCreator	an function or functor with the following signature boost::shared_ptr<RelogoLink> (Turtle* source, Turtle* target)

4.21.2.14 void repast::relogo::Turtle::createLinkWith (Turtle * *turtle*, const std::string & *network* = DEFAULT_UNDIR_NET)

Creates a link between this turtle and the specified turtle in the specified undirected network.

The network defaults to the default undirected network.

Parameters

turtle	the turtle to create the link with
network	the network to create the link in

4.21.2.15 template<typename LinkCreator > void repast::relogo::Turtle::createLinkWithLC (Turtle * *turtle*, LinkCreator & *creator*, const std::string & *network* = DEFAULT_UNDIR_NET)

Creates a link between this turtle and the specified turtle in the specified undirected network, using the specified linke creator.

The network defaults to the default undirected network.

Parameters

turtle	the turtle to create the link with
creator	the functor to create the link with
network	the network to create the link in

Template Parameters

LinkCreator	an function or functor with the following signature RelogoLink* (Turtle* source, Turtle* target)

4.21.2.16 void repast::relogo::Turtle::die ( )

Removes this turtle from the world.

Do not call this if there is a chance the Turtle will be referred to after the call to die()- for example, if it has moved and will be part of a move synchronization.

4.21.2.17 double repast::relogo::Turtle::distance (Turtle * *turtle*) const

Gets the distance from this turtle to the specified turtle.

Parameters

turtle	the turtle to get the distance to

Returns

the distance from this turtle to the specified turtle.

4.21.2.18 template<typename PatchType , typename ValueGetter > void repast::relogo::Turtle::downhill (ValueGetter & *getter*)

Moves this turtle to a neighboring patch with lowest value as retrieved via the ValueGetter.

The 8 neighboring patches and the current patch the turtle is on are considered. If no surrounding patch has a lower value than the patch the turtle is on, this turtle stays on the current patch. If there is more than one patch with the minimum value, then one will be chosen at random.Note that this turtle will end up in the center of one of the surrounding patches or in the center of its current patch.

Parameters

getter	the function or functor used to retrieve the value from the patch

Template Parameters

PatchType	the patch's type
a	functor or function with the following signature double (PatchType* patch) const

4.21.2.19 template<typename PatchType , typename ValueGetter > void repast::relogo::Turtle::downhill4 (ValueGetter & *getter*)

Moves this turtle to the patch with lowest value as retrieved via the ValueGetter.

The 4 neighboring patches and the current patch the turtle is on are considered. If no surrounding patch has a lower value than the patch the turtle is on, this turtle stays on the current patch. If there is more than on patch with the minimum value, then one will be chosen at random. This considers only the current patch and the 4 surrounding patches (N, S, E, W).

Note that this turtle will end up in the center of one of the surrounding patches or in the center of its current patch.

Parameters

getter	the function or functor used to retrieve the value from the patch

Template Parameters

PatchType	the patch's type
a	functor or function with the following signature double (PatchType* patch) const

4.21.2.20 double repast::relogo::Turtle::dx ( ) const

Gets the distance traveled along the x dimension if the turtle were to take one step forward along its current heading.

Returns

the distance traveled along the x dimension if the turtle were to take one step forward along its current heading.

4.21.2.21 double repast::relogo::Turtle::dy ( ) const

Gets the distance traveled along the y dimension if the turtle were to take one step forward along its current heading.

Returns

the distance traveled along the y dimension if the turtle were to take one step forward along its current heading.

4.21.2.22 void repast::relogo::Turtle::face (Turtle * *turtle*)

Sets the turtles heading to face towards the specified turtle.

Parameters

turtle	the turtle to face

4.21.2.23 void repast::relogo::Turtle::face (Patch * *patch*)

Sets the turtles heading to face towards the specified pach.

Parameters

patch	the patch to face

4.21.2.24 void repast::relogo::Turtle::facexy (double *nx,* double *ny*)

Sets the turtles heading to face the specified coordinates.

Parameters

nx	the x coordinate of the location to face
ny	the y coordinate of the location to face

4.21.2.25 void repast::relogo::Turtle::fd (double *distance*) [inline]

Moves this turtle forward the specified distance.

Parameters

distance	the distance to move

4.21.2.26 void repast::relogo::Turtle::forward (double *distance*) [inline]

Moves this turtle forward the specified distance.

Parameters

distance	the distance to move

4.21.2.27 float repast::relogo::Turtle::heading ( ) const [inline]

Gets this Turtle's current heading.

Returns

this Turtle's current heading.

4.21.2.28 void repast::relogo::Turtle::heading (float *heading*)

Sets this turtle's heading to the specified heading.

Parameters

heading	the new heading

4.21.2.29 boost::shared_ptr< RelogoLink > repast::relogo::Turtle::inLinkFrom (Turtle * *turtle,* const std::string & *name =* DEFAULT_DIR_NET)

Gets the link from the specified turtle to this one in the specified network which defaults to the default directed network.

Parameters

turtle	the turtle to get the link from
name	the name of the network containing the link

Returns

the link from the specified turtle to this one in the specified network which defaults to the default directed network.

4.21.2.30 bool repast::relogo::Turtle::inLinkNeighborQ (Turtle * *turtle,* const std::string & *name =* DEFAULT_DIR_NET)

Gets whether or not there is an edge into this turtle from the specified turtle, in the specified network.

The network defaults to the default directed network.

Parameters

turtle	the turtle to check as the source of the edge
name	the name of the network to check, defaults to the default directed network.

Returns

true if there is an edge into this turtle from the specified turtle in the named directed network, otherwise false.

4.21.2.31 template<typename AgentType > void repast::relogo::Turtle::inLinkNeighbors (AgentSet< AgentType > & *out,* const std::string & *name =* DEFAULT_DIR_NET)

Gets all the network predecessors of this turtle in the named network and puts them in the specified array list.

Parameters

out	the AgentSet to the return the neighbors in
name	the name of the network to get the network neighbors from

Template Parameters

AgentType	the type of agents to find in the network

4.21.2.32 void repast::relogo::Turtle::jump (double *distance*) [inline]

Moves this turtle forward the specified distance, if and only if that would not take this turtle outside the current topology.

Parameters

distance	the amount to move

4.21.2.33 void repast::relogo::Turtle::left (float *degrees*)

Turns the turtle left by the specified number of degrees.

To turn right, use a negative number.

Parameters

degrees	the amount to turn

4.21.2.34 bool repast::relogo::Turtle::linkNeighborQ (Turtle * *turtle,* const std::string & *name* = DEFAULT_UNDIR_NET)

Gets whether or not this turtle is linked to the specified turtle, in the specified network.

The network defaults to the default undirected network.

Parameters

turtle	the turtle to check that this links to
name	the name of the network to check, defaults to the default undirected network.

Returns

true if this this turtle is linked to the specified turtle in the named undirected network, otherwise false.

4.21.2.35 template<typename AgentType > void repast::relogo::Turtle::linkNeighbors (AgentSet< AgentType > & *out,* const std::string & *name* = DEFAULT_UNDIR_NET)

Gets all the network neighbors of this turtle in the named network and puts them in the specified AgentSet.

Parameters

out	the AgentSet to the return the neighbors in
name	the name of the network to get the network neighbors from

Template Parameters

AgentType	the type of agents to find in the network

4.21.2.36 boost::shared_ptr< RelogoLink > repast::relogo::Turtle::linkWith (Turtle * *turtle,* const std::string & *name* = DEFAULT_UNDIR_NET)

Gets the link between this turtle and the specified on in the named undirected network.

The network defaults to the default undirected network.

Returns

the link between this turtle and the specified on in the named undirected network.

4.21.2.37 void repast::relogo::Turtle::lt (float *degrees*)

Turns the turtle left by the specified number of degrees.

To turn right, use a negative number.

Parameters

degrees	the amount to turn

4.21.2.38 void repast::relogo::Turtle::move (double *distance*)

Moves this turtle the specified distance along the current heading.

Parameters

distance	the distance to move

4.21.2.39 void repast::relogo::Turtle::moveTo (Turtle * *turtle*)

Moves this turtle to the location of the specified turtle.

Parameters

turtle	the turtle whose location this turtle will be moved to

4.21.2.40 void repast::relogo::Turtle::moveTo (Patch * *patch*)

Moves this turtle to the location of the specified patch.

Parameters

patch	the patch whose location this turtle will be moved to

4.21.2.41 void repast::relogo::Turtle::mv (double *distance*) [inline]

Moves this turtle the specified distance along the current heading.

Parameters

distance	the distance to move

4.21.2.42 bool repast::relogo::Turtle::outLinkNeighborQ (Turtle * *turtle,* const std::string & *name* = DEFAULT_DIR_NET)

Gets whether or not there is an edge from this turtle to the specified turtle, in the specified network.

The network defaults to the default directed network.

Parameters

turtle	the turtle to check as the target of the edge
name	the name of the network to check, defaults to the default directed network.

Returns

true if there is an edge from this turtle into the specified turtle in the named directed network, otherwise false.

4.21.2.43 template<typename AgentType > void repast::relogo::Turtle::outLinkNeighbors (AgentSet< AgentType > & *out,* const std::string & *name* = DEFAULT_DIR_NET)

Gets all the network successors of this turtle in the named network and puts them in the specified array list.

Parameters

out	the AgentSet to the return the neighbors in
name	the name of the network to get the network neighbors from

Template Parameters

AgentType	the type of agents to find in the network

4.21.2.44 boost::shared_ptr< RelogoLink > repast::relogo::Turtle::outLinkTo (Turtle * *turtle*, const std::string & *name* = DEFAULT_DIR_NET)

Gets the link from the this turtle to the specified turtle in the specified network which defaults to the default directed network.

Parameters

turtle	the turtle to get the link to
name	the name of the network containing the link

Returns

the link from the this turtle to the specified turtle in the specified network which defaults to the default directed network.

4.21.2.45 template<typename PatchType > PatchType * repast::relogo::Turtle::patchHere ( ) const

Gets the patch under this turtle.

Template Parameters

the	type of the Patch

4.21.2.46 template<typename PatchType > PatchType * repast::relogo::Turtle::patchLeftAndAhead (float *angleInDegrees*, double *distance*)

Gets the patch that is the specified distance from this turtle, at the specified angle (turning left) from this turtle's heading.

Returns 0 if the patch would be outside of the world.

Parameters

angleInDegrees	the angle
distance	the distance

Template Parameters

PatchType	the type of the Patch

Returns

the patch that is the specified distance from this turtle, at the specified angle (turning left) from this turtle's heading or 0 if the patch would be outside of the world.

4.21.2.47 template<typename PatchType > PatchType * repast::relogo::Turtle::patchRightAndAhead (float *angleInDegrees*, double *distance*)

Gets the patch that is the specified distance from this turtle, in the specified degrees (turning right) from this turtle's heading.

Returns 0 if the patch would be outside of the world.

Parameters

angleInDegrees	the angle
distance	the distance

Template Parameters

PatchType	the type of the Patch

Returns

the patch that is the specified distance from this turtle, at the specified angle (turning right) from this turtle's heading or 0 if the patch would be outside of the world.

4.21.2.48 int repast::relogo::Turtle::pxCor ( ) const [virtual]

Gets the Patch x coordinate of this Turtle.

Returns

the patch x coordinate of this Turtle.

Implements repast::relogo::AbstractRelogoAgent.

4.21.2.49 int repast::relogo::Turtle::pyCor ( ) const [virtual]

Gets the Patch x coordinate of this Turtle.

Returns

the patch x coordinate of this Turtle.

Implements repast::relogo::AbstractRelogoAgent.

4.21.2.50 void repast::relogo::Turtle::setxy (double *x*, double *y*)

Sets the x and y coordinate of the Turtle's location.

If the location is outside of the world bounds this will throw an exception.

Parameters

x	the x coordinate
y	the y coordinate

4.21.2.51 float repast::relogo::Turtle::towards (RelogoAgent * *agent*) const

Gets the heading from this turtle to the specified RelogoAgent (turtle or patch).

Parameters

agent	the Turtle or Patch this will get the heading to

Returns

the heading from this turtle to the specified RelogoAgent (turtle or patch).

4.21.2.52 float repast::relogo::Turtle::towards (const Point< double > & *location*) const

Gets the heading from this turtle to the specified location.

Parameters

location	the location to get the heading to

Returns

the heading from this turtle to the specified location.

4.21.2.53 float repast::relogo::Turtle::towardsxy (double *x*, double *y*) const

Gets the heading from this turtle to the specified location.

Parameters

x	the x coordinate of the location
y	the y coordinate of the location

Returns

the heading from this turtle to the specified location.

4.21.2.54 template<typename PatchType , typename ValueGetter > void repast::relogo::Turtle::uphill (ValueGetter & *getter*)

Moves this turtle to the patch with highest value as retrieved via the ValueGetter.

The 8 neighboring patches and the current patch the turtle is on are considered. If no surrounding patch has a higher value than the patch the turtle is on, this turtle stays on the current patch. If there is more than on patch with the minimum value, then one will be chosen at random.

Note that this turtle will end up in the center of one of the surrounding patches or in the center of its current patch.

Parameters

getter	the function or functor used to retrieve the value from the patch

Template Parameters

PatchType	the patch's type
a	functor or function with the following signature double (PatchType* patch) const

4.21.2.55 template<typename PatchType , typename ValueGetter > void repast::relogo::Turtle::uphill4 (ValueGetter & *getter*)

Moves this turtle to the patch with highest value as retrieved via the ValueGetter.

The 4 neighboring patches and the current patch the turtle is on are considered. If no surrounding patch has a higher value than the patch the turtle is on, this turtle stays on the current patch. If there is more than on patch with

the minimum value, then one will be chosen at random. This considers only the current patch and the 4 surrounding patches (N, S, E, W).

Note that this turtle will end up in the center of one of the surrounding patches or in the center of its current patch.

Parameters

getter	the function or functor used to retrieve the value from the patch

Template Parameters

PatchType	the patch's type
a	functor or function with the following signature double (PatchType* patch) const

4.21.2.56 void repast::relogo::Turtle::xCor (double *x*)

Sets the x coordinate of the Turtle's location.

If the location is outside of the world bounds this will throw an exception.

Parameters

x	the new coordinate

4.21.2.57 void repast::relogo::Turtle::yCor (double *y*)

Sets the y coordinate of the Turtle's location.

If the location is outside of the world bounds this will throw an exception.

Parameters

y	the new y coordinate

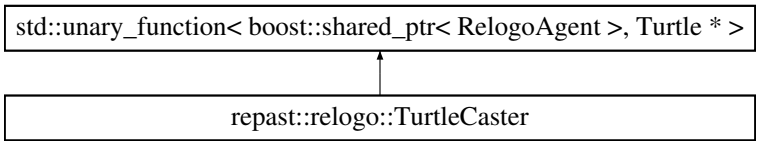
The documentation for this class was generated from the following files:

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Turtle.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Turtle.cpp

4.22 repast::relogo::TurtleCaster Struct Reference

Casts a pointer to a RelogoAgent to a pointer to a Turtle.

Inheritance diagram for repast::relogo::TurtleCaster:



Public Member Functions

- Turtle * **operator()** (boost::shared_ptr< RelogoAgent > ptr) const

4.22.1 Detailed Description

Casts a pointer to a RelogoAgent to a pointer to a Turtle.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Observer.cpp

4.23 repast::relogo::TypeInfoCmp Struct Reference

Compare two elements of type std::type_info using 'before'.

`#include <Observer.h>`

Public Member Functions

- bool **operator()** (const std::type_info *one, const std::type_info *two) const

4.23.1 Detailed Description

Compare two elements of type std::type_info using 'before'.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/Observer.h

4.24 repast::relogo::WorldCreator Class Reference

Creates a the relogo world given some parameters.

`#include <WorldCreator.h>`

Public Member Functions

- **WorldCreator** (boost::mpi::communicator *world)
- template<typename ObsType , typename PatchType , typename PatchCreator >
  ObsType * createWorld (const WorldDefinition &worldDef, const std::vector< int > &pConfig, PatchCreator &patchCreator)

  Creates the Relogo world using the specified parameters and returns an Observer of ObsType.
- template<typename ObsType , typename PatchType >
  ObsType * createWorld (const WorldDefinition &worldDef, const std::vector< int > &pConfig)

  Creates an observer of the specified type.

4.24.1 Detailed Description

Creates a the relogo world given some parameters.

4.24.2 Member Function Documentation

4.24.2.1 template<typename ObsType , typename PatchType , typename PatchCreator > ObsType * repast::relogo::WorldCreator::createWorld (const WorldDefinition & *worldDef*, const std::vector< int > & *pConfig*, PatchCreator & *patchCreator*)

Creates the Relogo world using the specified parameters and returns an Observer of ObsType.

Parameters

worldDef	the world definition
pConfig	a two element vector describing the number of processes along the x and y dimensions
patchCreator	used to create the Patches.

Template Parameters

ObsType	the type of Observer to create. This must extend Observer.
PatchType	the type of Patches to create. This must either be or extend Patch.
PatchCreator	a function or functor with the following signature PatchType* (AgentId id, Observer* obs).

4.24.2.2 template<typename ObsType , typename PatchType > ObsType * repast::relogo::WorldCreator::createWorld (const WorldDefinition & *worldDef,* const std::vector< int > & *pConfig*)

Creates an observer of the specified type.

The observer will contain a world defined by worldDef and pConfig.

Parameters

worldDef	the world definition
pConfig	a 2D vector containing the number of processes per grid dimension

Template Parameters

ObsType	the type of Observer to create. This must extend Observer.
PatchType	the type of Patches to create. This must either be or extend Patch.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/WorldCreator.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/WorldCreator.cpp

4.25 repast::relogo::WorldDefinition Class Reference

Defines a Relogo world.

`#include <WorldDefinition.h>`

Public Types

- typedef std::vector
  < Projection< RelogoAgent >
  * >::const_iterator proj_iter

  An iterator over pointers to Projection<RelogoAgent>.

Public Member Functions

- WorldDefinition (int minX, int minY, int maxX, int maxY, bool wrapped, int buffer)

  Creates a world definition with the specified parameters.
- void defineNetwork (std::string name, bool directed, RelogoLinkContentManager *rlcm)

  Defines a network with the specified name and whether or not the network is directed.
- void defineNetwork (bool directed, RelogoLinkContentManager *rlcm)

  Defines the default network and whether or not the network is directed.
- proj_iter networks_begin () const

  Gets the start of an iterator over the network Projections defined in this WorldDefinition.
- proj_iter networks_end () const

  Gets the end of an iterator over the network Projections defined in this WorldDefinition.

- int minX () const

  Gets the minimum x coordinate of the world.
- int minY () const

  Gets the minimum y coordinate of the world.
- int maxX () const

  Gets the maximum x coordinate of the world.
- int maxY () const

  Gets the maximum y coordinate of the world.
- const GridDimensions dimensions () const

  Gets the dimensions of the world expressed as a GridDimensions.
- bool isWrapped () const

  Gets whether or not the world wraps.
- int buffer () const

  Gets the size of the grid / space buffer.

4.25.1 Detailed Description

Defines a Relogo world.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 repast::relogo::WorldDefinition::WorldDefinition (int *minX,* int *minY,* int *maxX,* int *maxY,* bool *wrapped,* int *buffer*)

Creates a world definition with the specified parameters.

These parameter will be applied when the world is created using a WorldCreator.

Parameters

minX	the minimum x coordinate of the world
minY	the minimum y coordinate of the world
maxX	the maximum x coordinate of the world
maxY	the maximum y coordinate of the world
wrapped	whether or not the space is periodic, wrapped as a torus
buffer	the size of the grid and space buffer between process grid and space representations

4.25.3 Member Function Documentation

4.25.3.1 int repast::relogo::WorldDefinition::buffer ( ) const [inline]

Gets the size of the grid / space buffer.

Returns

the size of the grid / space buffer.

4.25.3.2 void repast::relogo::WorldDefinition::defineNetwork (std::string *name,* bool *directed,* RelogoLinkContentManager * *rlcm*)

Defines a network with the specified name and whether or not the network is directed.

The network will use the default RelogoEdge.

Parameters

name	the name of the network
directed	if true, the network will be directed, otherwise it will be undirected

4.25.3.3 void repast::relogo::WorldDefinition::defineNetwork (bool *directed,* RelogoLinkContentManager * *rlcm*)

Defines the default network and whether or not the network is directed.

Any network related calls that don't specify a name will use this network. The network will use RelogoEdge-s by default

Parameters

directed	if true, the network will be directed, otherwise it will be undirected

4.25.3.4 const GridDimensions repast::relogo::WorldDefinition::dimensions ( ) const [inline]

Gets the dimensions of the world expressed as a GridDimensions.

Returns

the dimensions of the world expressed as a GridDimensions.

4.25.3.5 bool repast::relogo::WorldDefinition::isWrapped ( ) const [inline]

Gets whether or not the world wraps.

Returns

true if the world wraps, otherwise false.

4.25.3.6 int repast::relogo::WorldDefinition::maxX ( ) const [inline]

Gets the maximum x coordinate of the world.

Returns

the maximum x coordinate of the world.

4.25.3.7 int repast::relogo::WorldDefinition::maxY ( ) const [inline]

Gets the maximum y coordinate of the world.

Returns

the maximum y coordinate of the world.

4.25.3.8 int repast::relogo::WorldDefinition::minX ( ) const [inline]

Gets the minimum x coordinate of the world.

Returns

the minimum x coordinate of the world.

4.25.3.9 int repast::relogo::WorldDefinition::minY ( ) const [inline]

Gets the minimum y coordinate of the world.

Returns

the minimum y coordinate of the world.

4.25.3.10 proj_iter repast::relogo::WorldDefinition::networks_begin ( ) const [inline]

Gets the start of an iterator over the network Projections defined in this WorldDefinition.

The iterator returns a pointer to a Projection<RelogoAgent>*.

4.25.3.11 proj_iter repast::relogo::WorldDefinition::networks_end ( ) const [inline]

Gets the end of an iterator over the network Projections defined in this WorldDefinition.

The iterator returns a pointer to a Projection<RelogoAgent>*.

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

- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/WorldDefinition.h
- /Users/murphy/work/RepastHPC_GIT/repast.hpc/src/relogo/WorldDefinition.cpp