ecosistem_1.0

1.0

Generated by Doxygen 1.7.3

Wed Mar 30 2011 17:14:51

Contents

Chapter 1

Ecosystem Simulator

1.1 Introduction

this project is a didactic project which aim is to model and simulate the evolution of an ecosystem.

1.2 Model Features

the ecosystem is developed to have the following features:

1.2.1 N Species

the ecosystem could have n species whith different parameters

1.2.2 Mutual Appetite

each species could like, and so eat, each other. selected a species the user can set how much a species like another. rabbit like carrot but doesn't like wolf.

1.2.3 Vivent Gender

vivent could have a gender: male, female, asexual, ermaphrodite. in this realize animals could only be male or female. no vegetable reproduction is modeled nor implemented so vegetable gender is only asexual;

1.3 Purposes

as already said this project has only didactic purposes. i can't really assure that it can produce good result from a scientific point of view. my real purpose was to develop something that could be expanded easily in future realizes, and of course have practice whith OO programming and boost features like multi index containers which are the key components of this project.

1.4 Vivent Model

vivent are modelized giving them differents parameters. you can see them by looking to the vivent's inheritance tree.

Chapter 2

Todo List

Member EcosystemContainer::is_full() implement

Member EcosystemContainer::step(StepLog &log) could be a good idea to make the whole couple migrate

Member SpeciesInfo::likings scrivi delle considerazioni finali sul fatto che i multi_-index sono più comodi in questi casi anche per emulare una map isi isi

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

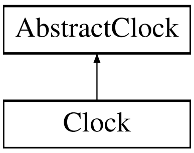
Class Documentation

6.1 AbstractClock Class Reference

abstract class for the clock

`#include <time.h>`

Inheritance diagram for AbstractClock:



Public Member Functions

- virtual void tick (unsigned int times=1)=0

  make the clock tick

6.1.1 Detailed Description

abstract class for the clock

Definition at line 53 of file time.h.

6.1.2 Member Function Documentation

6.1.2.1 virtual void AbstractClock::tick (unsigned int *times* = `1`) `[pure virtual]`

make the clock tick

abstract function

Implemented in Clock.

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

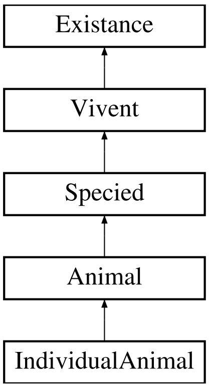
- sources/time.h
- sources/time.cpp

6.2 Animal Class Reference

class Animal

```
#include <animal.h>
```

Inheritance diagram for Animal:



Public Member Functions

- Animal (unsigned int u_fight_coast=1)

  default constructor

- ∼Animal ()

  default destructor

- virtual void live ()

  implementation of Specied::live()

- unsigned int & fight_coast ()

  set fight coast

- unsigned int fight_coast () const

  get fight coast

Private Attributes

- unsigned int m_fight_coast

  fight coast is the coast to pay everytime a fight occours

6.2.1 Detailed Description

class Animal this class rappresents the animal as a form of life able to move and eat. animals can eat other animals or vegetables. it depends from the liking between the species of the two form of lifes

Definition at line 37 of file animal.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Animal::Animal (unsigned int *u_fight_coast* = `1`)

default constructor

does nothing

Definition at line 35 of file animal.cpp.

6.2.2.2 Animal::∼Animal ( )

default destructor

does nothing

Definition at line 40 of file animal.cpp.

6.2.3 Member Function Documentation

6.2.3.1 void Animal::live ( ) `[virtual]`

implementation of Specied::live()

See also

Specied::live()

Definition at line 45 of file animal.cpp.

6.2.4 Member Data Documentation

6.2.4.1 unsigned int Animal::m_fight_coast `[private]`

fight coast is the coast to pay everytime a fight occours the pay coast is suctract from the hp both of the prey and the predator.

Definition at line 68 of file animal.h.

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

- sources/animal.h
- sources/animal.cpp

6.3 change_animal_appetite Struct Reference

functor to change IndividualAnimal appetite

`#include <fieldchangers.hpp>`

Public Member Functions

- change_animal_appetite (unsigned int nw_appetite)

  constructor setting new appetite value

- void operator() (IndividualAnimal &an)

  change the value

Private Attributes

- unsigned int m_nw_appetite

  new appetite value

6.3.1 Detailed Description

functor to change IndividualAnimal appetite

Definition at line 134 of file fieldchangers.hpp.

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

- sources/fieldchangers.hpp

6.4 change_animal_hp Struct Reference

functor to change IndividualAnimal hp

`#include <fieldchangers.hpp>`

Public Member Functions

- change_animal_hp (int &nw_hp)

  constructor setting new hp value

- void operator() (IndividualAnimal &an)

  change the value

Private Attributes

- int m_nw_hp

  new hp value

6.4.1 Detailed Description

functor to change IndividualAnimal hp this version controll if the hp passed are negative. if so set it to 0 if they'r more than 100 set it to 0 because of uncorrect usage of unsigned int

Definition at line 68 of file fieldchangers.hpp.

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

- sources/fieldchangers.hpp

6.5 change_animal_libido Struct Reference

functor to cahnge IndividualAnimal libido

```
#include <fieldchangers.hpp>
```

Public Member Functions

- change_animal_libido (unsigned int nw_libido)

  constructor setting new lidibo value

- void operator() (IndividualAnimal &an)

  change the value

Private Attributes

- unsigned int m_nw_libido

  new libido value

6.5.1 Detailed Description

functor to cahnge IndividualAnimal libido

Definition at line 114 of file fieldchangers.hpp.

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

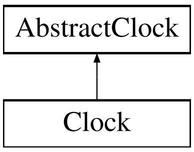
- sources/fieldchangers.hpp

6.6 Clock Class Reference

real clock able to give the time of the sistem

```
#include <time.h>
```

Inheritance diagram for Clock:



Public Member Functions

- Clock (long int u_abs=0, double u_rel=0)

  constructor giving initial conditions

- virtual void tick (unsigned int times=1)

  make the clock tick

- void calculate_relative ()

  calculate the relative time

- long int & abs ()

  set abs

- double & rel ()

  set rel

- const long int & abs () const

  get abs

- const double & rel () const

  get rel

Private Attributes

- long int m_absolute_time

  absolute time

- double m_relative_time

  relative time

Friends

- std::ostream & operator<< (std::ostream &os, const Clock &clock)

  prints the time

6.6.1 Detailed Description

real clock able to give the time of the sistem give the absolute and relative time of the sistem

Definition at line 68 of file time.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Clock::Clock (long int *u_abs* = `0`, double *u_rel* = `0`)

constructor giving initial conditions

construcor giving the initial absolute time and relative time. default is 0,0

Parameters

u_abs	starting absolute time
u_rel	starting relative time

Definition at line 46 of file time.cpp.

6.6.3 Member Function Documentation

6.6.3.1 void Clock::tick (unsigned int *times* = `1`) `[virtual]`

make the clock tick

increase the m_absolute time and calculate the relative time

Implements AbstractClock.

Definition at line 69 of file time.cpp.

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

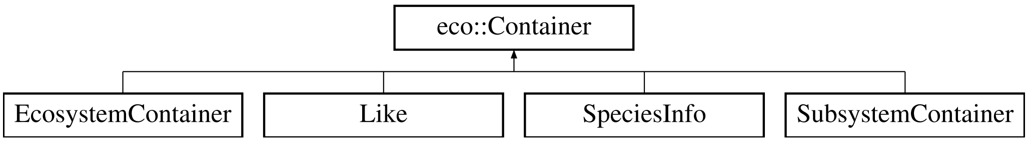
- sources/time.h
- sources/time.cpp

6.7 eco::Container Class Reference

Container abstract class.

```
#include <container.h>
```

Inheritance diagram for eco::Container:



Public Member Functions

- Container ()

  default constructor

- ~Container ()

  default destructor

- virtual bool is_full ()=0

  is the container full

6.7.1 Detailed Description

Container abstract class.

Definition at line 30 of file container.h.

6.7.2 Member Function Documentation

6.7.2.1 bool Container::is_full ( ) [pure virtual]

is the container full

abstract member

Implemented in EcosystemContainer, Like, SpeciesInfo, and SubsystemContainer.

Definition at line 38 of file container.cpp.

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

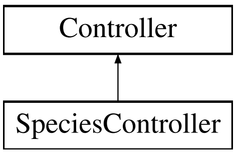
- sources/container.h
- sources/container.cpp

6.8 Controller Class Reference

class that generally controll

`#include <controller.h>`

Inheritance diagram for Controller:



Public Member Functions

- Controller ()

  default constructor

- ~Controller ()

  default destructor

- virtual bool check ()=0

  check what is to controll

6.8.1 Detailed Description

class that generally controll

Definition at line 33 of file controller.h.

6.8.2 Member Function Documentation

6.8.2.1 virtual bool Controller::check ( ) `[pure virtual]`

check what is to controll

abstarct function

Implemented in SpeciesController.

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

- sources/controller.h
- sources/controller.cpp

6.9 DateOfBirth Class Reference

simple class for the date of birh

```
#include <time.h>
```

Public Member Functions

- DateOfBirth (long int u_abs=0, double u_rel=0)

  default creator
- DateOfBirth (const Clock &u_clock)

  creator whith a clock
- ∼DateOfBirth ()

  default destructor
- const long int & abs () const

  get abs
- long int & abs ()

  set abs
- const double & rel () const

  get rel
- double & rel ()

  set rel

Private Attributes

- long int m_absolute

  absolute date of birth
- double m_relative

  relative date of birth

Friends

- std::ostream & operator<< (std::ostream &os, const DateOfBirth &date)

  stream to print the DateOfBirth

6.9.1 Detailed Description

simple class for the date of birh this class is a simple wrapper that contain the information relatives to the date of life of the form of life. it contains the relative and the absolute date of birth

Definition at line 122 of file time.h.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 DateOfBirth::DateOfBirth (long int *u_abs* = `0`, double *u_rel* = `0`)

default creator

gives all 0 value, not assign clock_ref

Definition at line 100 of file time.cpp.

6.9.2.2 DateOfBirth::DateOfBirth (const Clock & *u_clock*)

creator whith a clock

set the birth date using the clock you pass it

Definition at line 108 of file time.cpp.

6.9.3 Member Data Documentation

6.9.3.1 long int DateOfBirth::m_absolute `[private]`

absolute date of birth

date of birth expressed as the number of calls occurred before the creation of the form of life

Definition at line 162 of file time.h.

6.9.3.2 double DateOfBirth::m_relative `[private]`

relative date of birth

date of birth expressed as number of cicle occurred before the creation of the form of life

Definition at line 167 of file time.h.

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

- sources/time.h
- sources/time.cpp

6.10 SubsystemContainer::eat Struct Reference

boost multyindex::ordered_index tag

```
#include <subsystemcontainer.h>
```

6.10.1 Detailed Description

boost multyindex::ordered_index tag

Definition at line 83 of file subsystemcontainer.h.

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

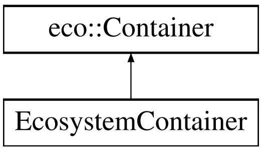
- sources/subsystemcontainer.h

6.11 EcosystemContainer Class Reference

contains all the form of life

```
#include <ecosystem.h>
```

Inheritance diagram for EcosystemContainer:



Public Types

- typedef boost::multi_array< SubsystemContainer, 2 > ecosys_type

  the type of the ecosystem is a boost multi_array container

Public Member Functions

- EcosystemContainer (unsigned int u_x_size=0, unsigned int u_y_size=0, bool u_bound=true, unsigned int u_species_number=1, bool u_random_seed=false) *default constructor*

- ~EcosystemContainer ()

  default destructor

- virtual bool is_full ()

  is the system full?

- void initialize (std::istream &is)

  initialize the sistem for the step()

- void initialize (std::ifstream &infs)

  initialize the sistem for step()

- bool insert (IndividualAnimal &u_an, const int u_x, const int u_y)

  insert animal in subecosystem container

- bool insert (IndividualVegetable &u_veg, const int u_x, const int u_y)

  insert vegetable in the ecosystem container

- bool step ()

  step evolution

- bool step (StepLog &log)

  step evoultion producing steplog

- void FillRandom ()

  fill random ecosystem

- bool fill (std::ifstream &ifs)

  fill the ecosystem from a file

- std::pair< unsigned int, bool > specied_population (const unsigned int u_spec_id=0)

  total number of specied of this species

- void draw_evolv (const unsigned int steps)

  draw and evolv

- void draw_evolv_fast (const unsigned int steps, std::string options=std::string("refresh_populations"))

  draw the evolution of the system in a fast way

- bool evolv (unsigned int steps)

  make the system evolv

- bool evolv (unsigned int steps, std::vector< StepLog > &logs)

  make the system evolv and create a log

- bool evolv (unsigned int steps, std::ofstream &ofs)

  evolv the system and write logs

- ecosys_type & ecosystem ()

  set the ecosystem

- SubsystemContainer & subsystem (const int u_x, const int u_y)

  set the SubsystemContainer

- const ecosys_type & ecosystem () const

  get the ecosystem

- const SubsystemContainer & subsystem (int u_x, int u_y) const

  get the SubsystemContainer

Private Member Functions

- std::pair< bool, bool > m_newborn (const unsigned int species_id, const unsigned int subs_x, const unsigned int subs_y, unsigned int u_hp)

  create a newborn for the species indicated

- std::pair< bool, bool > m_newborn (const unsigned int species_id, const unsigned int subs_x, const unsigned int subs_y, unsigned int u_hp, StepLog &log)

  create a newborn for the species indicated

- int m_rand_int (int a=0, int b=0)

  get a random int

- void m_dead (SubsystemContainer::an_eat_it dead_an, SubsystemContainer::index_an_by_eat &eat_index)

  remove animal from the index provided

- void m_dead (SubsystemContainer::an_eat_it dead_an, SubsystemContainer::index_an_by_eat &eat_index, StepLog &log, int x, int y)

  remove animal from the index provided and create a log

- void m_dead (SubsystemContainer::an_id_it dead_an, SubsystemContainer::index_an_by_id &id_index)

  remove animal from the index provided

- void m_dead (SubsystemContainer::an_id_it dead_an, SubsystemContainer::index_an_by_id &id_index, StepLog &log, int x, int y) *remove animal from the index provided and create a log*

- void m_dead (SubsystemContainer::an_reproduce_it dead_an, SubsystemContainer::index_an_by_reproduce &repr_index)

  remove animal from the index provided

- void m_dead (SubsystemContainer::an_reproduce_it dead_an, SubsystemContainer::index_an_by_reproduce &repr_index, StepLog &log, int x, int y)

  remove animal from the index provided and create a log

- void m_bound_translator (int &x, int &y)

  translate the coordinate if there were no boundaries

- bool m_where_to_move (int &u_x, int &u_y, const int curr_x, const int curr_y, const unsigned int species_id)

  where an animal could move

- bool m_migrate (const int curr_x, const int curr_y, SubsystemContainer::an_id_it &to_move, SubsystemContainer::index_an_by_id &idx)

  migrate form curr_x and curr_y using m_where_to_move

- bool m_migrate (const int curr_x, const int curr_y, SubsystemContainer::an_id_it &to_move, SubsystemContainer::index_an_by_id &idx, StepLog &log)

  migrate form curr_x and curr_y using m_where_to_move

Private Attributes

- unsigned int m_x_size

  x size of the container

- unsigned int m_y_size

  y size of the container

- bool m_boundaries

  if the ecosystem have boundaries or not

- ecosys_type m_ecosystem

  the ecosystem as ensable of subsystem containers

- unsigned long int m_last_existance_id

  the last existance id of a new creature

- SpeciesController m_species_controller

  the species controller for this ecosys

- Clock m_clock

  the clock of the system

- boost::mt19937 m_generator

  the random number generator used by m_rand_int

Friends

- std::ifstream & **operator>>** (std::ifstream &is, EcosystemContainer &eco)

6.11.1 Detailed Description

contains all the form of life

Definition at line 49 of file ecosystem.h.

6.11.2 Member Typedef Documentation

6.11.2.1 typedef boost::multi_array<SubsystemContainer, 2> EcosystemContainer::ecosys_type

the type of the ecosystem is a boost multi_array container

roughly is a matrix

Definition at line 55 of file ecosystem.h.

6.11.3 Member Function Documentation

6.11.3.1 void Eco::draw_evolv (const unsigned int *steps*)

draw and evolv

evolv the sistem for a number of steps and draw his evolution this method is slower than draw_evolv_fast because for every step it reset the TH2F and compute the total population for every species in every single subsystem. although the rappresentation is always right.

Definition at line 61 of file draw_evolv.cpp.

6.11.3.2 void Eco::draw_evolv_fast (const unsigned int *steps*, std::string *options* = `std::string("refresh_populations")`)

draw the evolution of the system in a fast way

the system will evolv and for each step will be draw something according to options parameter. this method is faster than draw_evolv because it parses the log produced by step(StepLog) and modify the rappresentation whithout performing a query for every subsystem and for every species. it doesn't reset the TH2F but only modify the wheight of the interested bins

Definition at line 352 of file draw_evolv.cpp.

6.11.3.3 bool Eco::evolv (unsigned int *steps*)

make the system evolv

Parameters

steps	number times step() will be runned

Returns

false if step fails.

Definition at line 1362 of file ecosystem.cpp.

6.11.3.4 bool Eco::evolv (unsigned int *steps*, std::vector< StepLog > & *logs*)

make the system evolv and create a log

Parameters

steps	number times step() will be runned
Logs	a vector of StepLog

Returns

false if step fails.

See also

StepLog

Definition at line 1384 of file ecosystem.cpp.

6.11.3.5 bool Eco::evolv (unsigned int *steps*, std::ofstream & *ofs*)

evolv the system and write logs

evolv the system and once finisched or once step() returns false; write using the ofs to a file the results. results are elaborations of StepLog

Parameters

steps	number times step() will be runned
ofs	output file stream in which buffer

Returns

false if step fails.

Definition at line 1411 of file ecosystem.cpp.

6.11.3.6 void Eco::FillRandom ( )

fill random ecosystem

each subsystem will be filled by a random number of vivent for each species whith random hp and random sex. the distribution is uniform and linear. this filling method respect the SubsystemContainer::is_full.

this method is written ad FillRandom and not fillrandom just to do a tribute to root-cern lib.

Definition at line 357 of file ecosystem.cpp.

6.11.3.7 void Eco::initialize (std::istream & *is*)

initialize the sistem for the step()

Parameters

is	generical input stream

Returns

true if initialization suceed, false if fails

Definition at line 125 of file ecosystem.cpp.

6.11.3.8 void Eco::initialize (std::ifstream & *infs*)

initialize the sistem for step()

Parameters

infs	input file stream

Returns

true if initialization suceed, false if fails

Definition at line 173 of file ecosystem.cpp.

6.11.3.9 bool Eco::insert (IndividualAnimal & *u_an*, const int *u_x*, const int *u_y*)

insert animal in subecosystem container

Parameters

u_an	the animal to inser
u_x	the x coord of the subsystem you want to insert in
u_y	the y coord of the subsystem you want to insert in

Returns

true if insertion succed, false if fails

Definition at line 243 of file ecosystem.cpp.

6.11.3.10 bool Eco::insert (IndividualVegetable & *u_veg*, const int *u_x*, const int *u_y*)

insert vegetable in the ecosystem container

Parameters

u_veg	the vegetable to insert
u_x	the x coord of the subsystem you want to insert in
u_y	the y coord of the subsystem you want to insert in

Returns

true if insertion succed, false if fails

Definition at line 300 of file ecosystem.cpp.

6.11.3.11 bool Eco::is_full ( ) `[virtual]`

is the system full?

Todo

implement

Implements eco::Container.

Definition at line 92 of file ecosystem.cpp.

6.11.3.12 void Eco::m_bound_translator (int & *x*, int & *y*) `[private]`

translate the coordinate if there were no boundaries

if there were no boundaries the x and y coord could be more than m_x_size or m_y_size or less than 0. this function provide a translation of such numbers in an interval from 0 - m_x/y_size.

if the m_boundaries flag is not true and the coordinates are not in the interval it will print an error message and set both x and y to 0 example: if x = m_x_size+1 it becomes 0

Definition at line 794 of file ecosystem.cpp.

6.11.3.13 void Eco::m_dead (SubsystemContainer::an_id_it *dead_an*, SubsystemContainer::index_an_by_id & *id_index*) [private]

remove animal from the index provided

Parameters

dead_an	the animal
id_index	the id index of the SubsystemContainer::vegetable_set

Definition at line 719 of file ecosystem.cpp.

6.11.3.14 void Eco::m_dead (SubsystemContainer::an_eat_it *dead_an*, SubsystemContainer::index_an_by_eat & *eat_index*) [private]

remove animal from the index provided

Parameters

dead_an	the animal
eat_index	the eat index of the SubsystemContainer::vegetable_set

Definition at line 683 of file ecosystem.cpp.

6.11.3.15 void Eco::m_dead (SubsystemContainer::an_reproduce_it *dead_an*, SubsystemContainer::index_an_by_reproduce & *repr_index*) [private]

remove animal from the index provided

Parameters

dead_an	the animal
id_index	the id index of the SubsystemContainer::vegetable_set

Definition at line 755 of file ecosystem.cpp.

6.11.3.16 void Eco::m_dead (SubsystemContainer::an_eat_it *dead_an*, SubsystemContainer::index_an_by_eat & *eat_index*, StepLog & *log*, int *x*, int *y*) [private]

remove animal from the index provided and create a log

Parameters

dead_an	the animal
eat_index	the eat index of the SubsystemContainer::vegetable_set
log	insert a PopulationVariation inside the StepLog

x	the x coord of the subsystem in which the anima is
y	the y coord of the subsystem in which the anima is

Definition at line 696 of file ecosystem.cpp.

6.11.3.17 void Eco::m_dead (SubsystemContainer::an_id_it *dead_an*, SubsystemContainer::index_an_by_id & *id_index*, StepLog & *log*, int *x*, int *y*) [private]

remove animal from the index provided and create a log

Parameters

dead_an	the animal
id_index	the id index of the SubsystemContainer::vegetable_set
log	insert a PopulationVariation inside the StepLog
x	the x coord of the subsystem in which the anima is
y	the y coord of the subsystem in which the anima is

Definition at line 732 of file ecosystem.cpp.

6.11.3.18 void Eco::m_dead (SubsystemContainer::an_reproduce_it *dead_an*, SubsystemContainer::index_an_by_reproduce & *repr_index*, StepLog & *log*, int *x*, int *y*) [private]

remove animal from the index provided and create a log

Parameters

dead_an	the animal
id_index	the id index of the SubsystemContainer::vegetable_set
log	insert a PopulationVariation inside the StepLog
x	the x coord of the subsystem in which the anima is
y	the y coord of the subsystem in which the anima is

Definition at line 769 of file ecosystem.cpp.

6.11.3.19 bool Eco::m_migrate (const int *curr_x*, const int *curr_y*, SubsystemContainer::an_id_it & *to_move*, SubsystemContainer::index_an_by_id & *idx*) [private]

migrate form curr_x and curr_y using m_where_to_move

this method is called inside step. and move the animal from a subsystem to other.

this method were called in step() when:

- there is no food in the current subsystem

- there is no one for reproduction
- there is no space for a newborn

Parameters

curr_x	the current x subsystem coordinate
to_move	an iterator to the animal to move
idx	the current id_index in which the animal is. this parameter is necessary because the animal had to be removed from the current ecosystem

Returns

true if migration occours, else false

Definition at line 1111 of file ecosystem.cpp.

6.11.3.20 bool Eco::m_migrate (const int *curr_x*, const int *curr_y*, SubsystemContainer::an_id_it & *to_move*, SubsystemContainer::index_an_by_id & *idx*, StepLog & *log*) [private]

migrate form curr_x and curr_y using m_where_to_move

this method is called inside step. and move the animal from a subsystem to other.

this method were called in step() when:

- there is no food in the current subsystem
- there is no one for reproduction
- there is no space for a newborn

Parameters

curr_x	the current x subsystem coordinate
to_move	an iterator to the animal to move
idx	the current id_index in which the animal is. this parameter is necessary because the animal had to be removed from the current ecosystem
log	create a PopulationVariation and insert it in the the StepLog passed.

Returns

true if migration occours, else false

Definition at line 1215 of file ecosystem.cpp.

6.11.3.21 std::pair< bool, bool > Eco::m_newborn (const unsigned int *species_id*, const unsigned int *subs_x*, const unsigned int *subs_y*, unsigned int *u_hp*, StepLog & *log*) `[private]`

create a newborn for the species indicated

this function is used inside step()

Parameters

species_id	the species id of the newborn
subs_x	the x coord of the subsystem in which the newborn should be
subs_y	the y coord of the subsystem in which the newborn should be
log	the StepLog

Returns

first false if insertion fail, second false if there is no space for the newborn

See also

StepLog

Definition at line 933 of file ecosystem.cpp.

6.11.3.22 std::pair< bool, bool > Eco::m_newborn (const unsigned int *species_id*, const unsigned int *subs_x*, const unsigned int *subs_y*, unsigned int *u_hp*) `[private]`

create a newborn for the species indicated

this function is used inside step()

Parameters

species_id	the species id of the newborn
subs_x	the x coord of the subsystem in which the newborn should be
subs_y	the y coord of the subsystem in which the newborn should be
log	the StepLog

Returns

first false if insertion fail, second false if there is no space for the newborn

Definition at line 846 of file ecosystem.cpp.

6.11.3.23 int Eco::m_rand_int (int *a* = `0`, int *b* = `0`) `[private]`

get a random int

create a random integet from a to b included. the sistribution is uniform. boost random numbers generator where used

Definition at line 671 of file ecosystem.cpp.

6.11.3.24 bool Eco::m_where_to_move (int & *u_x,* int & *u_y,* const int *curr_x,* const int *curr_y,* const unsigned int *species_id*) [private]

where an animal could move

for a determinate species it controll inside near subsystem if they were full and took a random one of unfull

Parameters

u_x	the x coord animal will move
curr_x	the current x coord in which the animal is

Returns

false if there is no place to move

Definition at line 1031 of file ecosystem.cpp.

6.11.3.25 std::pair< unsigned int, bool > Eco::specied_population (const unsigned int *u_spec_id* = 0)

total number of specied of this species

Returns

a pair whith at first member the number of animals present in the ecosystem and at second member true if the species exists and false if not

Definition at line 1335 of file ecosystem.cpp.

6.11.3.26 bool Eco::step (StepLog & *log*)

step evoultion producing steplog

make a step in the evolution of the system and produce log.

this function in used in draw_evolv_fast

See also

StepLog
EcosystemContainer::draw_evolv_fast

Todo

could be a good idea to make the whole couple migrate

Definition at line 39 of file step_log.cpp.

6.11.3.27 bool Eco::step ( )

step evolution

make a step in the evolution of the system

Definition at line 39 of file step.cpp.

6.11.3.28 SubsystemContainer & Eco::subsystem (const int *u_x*, const int *u_y*)

set the SubsystemContainer

Parameters

u_x	x coordinate of the subsystem
u_y	y coordinate of the subsystem

Definition at line 556 of file ecosystem.cpp.

6.11.3.29 const SubsystemContainer & Eco::subsystem (int *u_x*, int *u_y*) const

get the SubsystemContainer

Parameters

u_x	x coordinate of the subsystem
u_y	y coordinate of the subsystem

Definition at line 618 of file ecosystem.cpp.

6.11.4 Member Data Documentation

6.11.4.1 ecosys_type EcosystemContainer::m_ecosystem [private]

the ecosystem as ensable of subsystem containers

as you can see in ecosys_type this is a bidimensional boost::multy_array (a matrix). please read the boost::multy_array doc befor to edit. range goes from [0][0] to [m_x_size-1][m_y_size-1]

Definition at line 422 of file ecosystem.h.

6.11.4.2 unsigned long int EcosystemContainer::m_last_existance_id [private]

the last existance id of a new creature

due to the fact that there is an unique id_number all the vivent inside a subsystem container vegetable or animal set every new vivent created can not have the same id of another. if this occurs the insertion in the subsystem will fail. so every time an animal is inserted or created by FillRandom() or m_newborn this variable is incremented;

Definition at line 432 of file ecosystem.h.

6.11.4.3 SpeciesController EcosystemContainer::m_species_controller [private]

the species controller for this ecosys

the species controller is initializiated calling the methods initialize.

Definition at line 438 of file ecosystem.h.

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

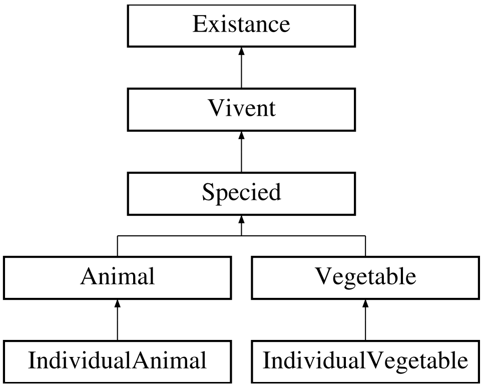
- sources/ecosystem.h
- sources/draw_evolv.cpp
- sources/ecosystem.cpp
- sources/step.cpp
- sources/step_log.cpp

6.12 Existance Class Reference

class Existance the most abstracted object

`#include <existance.h>`

Inheritance diagram for Existance:



Public Types

- typedef long unsigned int id_type

  the type of the id_number

Public Member Functions

- Existance (DateOfBirth u_birth_date=DateOfBirth(0, 0), id_type u_id_number=0)

  default constructor
- ∼Existance ()

  default destructor
- virtual bool is_alive ()=0

  is the existance alive?
- DateOfBirth & birth_date ()

  the birth date
- id_type & id_number ()

  the id numer
- const DateOfBirth & birth_date () const

  get the birth date
- id_type id_number () const

  get the id_number

Private Attributes

- DateOfBirth m_birth_date

  when an Existace is created
- id_type m_id_number

  unique identifier of a form of life

6.12.1 Detailed Description

class Existance the most abstracted object this class is the most abstract object and it's pure virtual class. all the form of life in the program eredit from her

Definition at line 40 of file existance.h.

6.12.2 Member Typedef Documentation

6.12.2.1 typedef long unsigned int Existance::id_type

the type of the id_number

long unsigned int

Definition at line 46 of file existance.h.

6.12.3 Member Function Documentation

6.12.3.1 DateOfBirth & Existance::birth_date ( )

the birth date

da birth date

See also

DateOfBirth
m_birth_date

Definition at line 48 of file existance.cpp.

6.12.3.2 Existance::id_type & Existance::id_number ( )

the id numer

See also

m_id_number

Definition at line 53 of file existance.cpp.

6.12.3.3 virtual bool Existance::is_alive ( ) [pure virtual]

is the existance alive?

returns true if the object is alive. existance cannot be whithout their specifications, so this member is pure virtual

Implemented in IndividualAnimal, Specied, and Vivent.

6.12.4 Member Data Documentation

6.12.4.1 DateOfBirth Existance::m_birth_date [private]

when an Existace is created

every form of life has a date of birth. see DateOfBirth for more info.

See also

DateOfBirth

Definition at line 88 of file existance.h.

6.12.4.2 id_type Existance::m_id_number `[private]`

unique identifier of a form of life

this data member is different between all the form of life istanced. this property has to be granted by the gestion algorithm

Definition at line 96 of file existance.h.

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

- sources/existance.h
- sources/existance.cpp

6.13 Gender Class Reference

the gender of the form of life

`#include <gender.h>`

Public Member Functions

- Gender ()

  default constructor

- Gender (std::string r_gender)

  constructor using a string

- ~Gender ()

  default destructor

- bool is_male ()

  returns true if it is male

- bool is_female ()

  returns true if it is female

- bool is_hermaphrodite ()

  returns true if it is hermaphrodite

- bool is_asexual ()

  returns true if it is asexual

- const std::string & gender () const

  get gender name

- void change_gender (std::string r_gender) *change the sex*

- void change_gender (const unsigned int u_gender_num=4)

  change the sex

- unsigned int numerical_gender () const

  numerical gender

- bool operator< (const Gender &gen) const

  operator <

Private Attributes

- bool male_

  if true is a male

- bool female_

  if true is a female

- std::string gender_name_

  name of the gender

Friends

- std::ostream & operator<< (std::ostream &os, const Gender &gen)

  ostream operator of gender

6.13.1 Detailed Description

the gender of the form of life the possible gender of a form of life where: male , female, ermaphrodite, asexual;

this is decided by the values of male_ and female_;

the ermaphrodite is male and female at the same time, the asexual is nor male nor female.

See also

male_
female_

Definition at line 48 of file gender.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Gender::Gender ( )

default constructor

if no argument were given the form of life is considered asexual

Definition at line 32 of file gender.cpp.

6.13.2.2 Gender::Gender (std::string *r_gender*)

constructor using a string

Parameters

r_gender	is the string containing the gender specification given in runtime. possible values are: "male" , "female" , "ermaphrodite" , "asexual";

if the gender is speciefied badly the gender is set to asexual;

Definition at line 42 of file gender.cpp.

6.13.3 Member Function Documentation

6.13.3.1 void Gender::change_gender (std::string *r_gender*)

change the sex

need a string like the constructos

Parameters

r_gender	see Gender()

Definition at line 110 of file gender.cpp.

6.13.3.2 void Gender::change_gender (const unsigned int *u_gender_num* = 4)

change the sex

Parameters

u_gender_num	see numerical_gender

Definition at line 152 of file gender.cpp.

6.13.3.3 unsigned int Gender::numerical_gender ( ) const

numerical gender

return a numerical id (int) which rappresent the gender. 1 is male, 2 is female, 3 is hermaphrodite and 4 is asexual;

Definition at line 195 of file gender.cpp.

6.13.4 Friends And Related Function Documentation

6.13.4.1 std::ostream& operator<< (std::ostream & *os*, const Gender & *gen*) [friend]

ostream operator of gender

prints a string saying the actual gender is a wrapper of Gender::gender()

Definition at line 213 of file gender.cpp.

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

- sources/gender.h
- sources/gender.cpp

6.14 Grafico Class Reference

Public Member Functions

- **Grafico** (string)
- void **Set_title** (string)
- void **Set_xlabel** (string)
- void **Set_ylabel** (string)
- void **Set_labels** (string, string)
- void **Set_xrange** (double, double)
- void **Set_yrange** (double, double)
- void **Add_grafico** (double ∗, double ∗, int, string)
- void **Set_data_style** (string)
- void **Legend_position** (int)

Private Attributes

- fstream **_comandi**
- char ∗ **_file**
- int **_N_grafici**
- string **_immagine**

6.14.1 Detailed Description

Definition at line 34 of file grafico.hpp.

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

- sources/grafico.hpp
- sources/grafico.cpp

6.15 SubsystemContainer::id Struct Reference

boost multyindex::ordered_index tag

```
#include <subsystemcontainer.h>
```

6.15.1 Detailed Description

boost multyindex::ordered_index tag

Definition at line 81 of file subsystemcontainer.h.

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

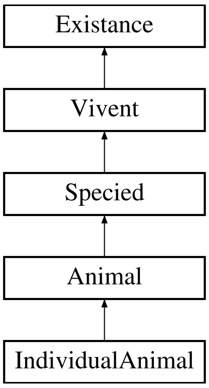
- sources/subsystemcontainer.h

6.16 IndividualAnimal Class Reference

class IndividualAnimal

```
#include <individualanimal.h>
```

Inheritance diagram for IndividualAnimal:



Public Member Functions

- IndividualAnimal (unsigned int u_appetite=0, unsigned int u_libido=0)

  default constructor

- ~IndividualAnimal ()

  default destructor

- virtual bool is_alive ()

  compute id the animal is alive

- unsigned int & appetite ()

  set appetite

- unsigned int & libido ()

  set libido

- unsigned int appetite () const

  get appetite

- unsigned int libido () const

  get libido

Private Attributes

- unsigned int m_appetite

  appetite factor

- unsigned int m_libido

  libido factor

Friends

- std::ostream & operator<< (std::ostream &os, const IndividualAnimal &an)

  ostream operator

6.16.1 Detailed Description

class IndividualAnimal this class rappresent the animal see as the final and real living form of life. each animal differs from the other by his appetite and his libido

Definition at line 39 of file individualanimal.h.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 IndividualAnimal::IndividualAnimal (unsigned int *u_appetite* = `0`, unsigned int *u_libido* = `0`)

default constructor

set appetite and libido, if none were given they were setted to 0 both.

Definition at line 37 of file individualanimal.cpp.

6.16.2.2 IndividualAnimal::∼IndividualAnimal ( )

default destructor

does nothing

Definition at line 44 of file individualanimal.cpp.

6.16.3 Friends And Related Function Documentation

6.16.3.1 std::ostream& operator<< (std::ostream & *os*, const IndividualAnimal & *an*) `[friend]`

ostream operator

prints al main info of the animal

Definition at line 76 of file individualanimal.cpp.

6.16.4 Member Data Documentation

6.16.4.1 unsigned int IndividualAnimal::m_appetite `[private]`

appetite factor

it rappresents the propensity of the animal to eat and rise when the hp of the animal were under the health status

See also

m_health_status

Definition at line 85 of file individualanimal.h.

6.16.4.2 unsigned int IndividualAnimal::m_libido `[private]`

libido factor

pay attention: parental controll pending. btw the libido factor determines the propensity of the animal to reproduce and rises when the animal's hp where under the health status

See also

m_health_status

Definition at line 93 of file individualanimal.h.

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

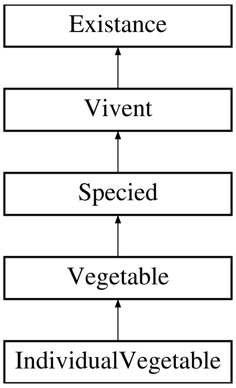
- sources/individualanimal.h
- sources/individualanimal.cpp

6.17 IndividualVegetable Class Reference

class IndividualVegetable

`#include <individualvegetable.h>`

Inheritance diagram for IndividualVegetable:



Public Member Functions

- IndividualVegetable ()

  default contructor

- ∼IndividualVegetable ()

  default destructor

Friends

- std::ostream & operator<< (std::ostream &os, const IndividualVegetable &veg)

  prints the individual vegetable

6.17.1 Detailed Description

class IndividualVegetable class rappresenting the vegetable seen as a single form of life. for this implementation no more features are added in order to differentiate IndividualVegetable to Specied. by the way this had been done to give the future possibiliti to implement vegetable reproduction or feeding.

Definition at line 40 of file individualvegetable.h.

6.17.2 Friends And Related Function Documentation

6.17.2.1 std::ostream& operator<< (std::ostream & *os*, const IndividualVegetable & *veg*) `[friend]`

prints the individual vegetable

prints id_number , species_id and hp

Definition at line 46 of file individualvegetable.cpp.

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

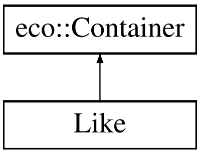
- sources/individualvegetable.h
- sources/individualvegetable.cpp

6.18 Like Struct Reference

how much a species is liked by another

`#include <like.h>`

Inheritance diagram for Like:



Public Member Functions

- Like (unsigned int u_spec_id=0, int u_like_factor=0)

    default constructor

- bool is_full ()

    is the like full

- bool operator< (const Like &lk)

Like were sorted by ascendent spec_id.

- bool operator> (const Like &lk)

  Like were sorted by ascendent spec_id.

- bool operator== (const Like &lk)

  egual operator

- bool operator!= (const Like &lk)

  not equal operator

Public Attributes

- unsigned int liked_spec_id

  the species id liked

- int like_factor

  how much the species is liked

Friends

- std::ostream & operator<< (std::ostream &os, const Like &like)

  print liked_species_id and like_factor

6.18.1 Detailed Description

how much a species is liked by another the liked_spec_id is the species id this species like and the like factor is how much he likes it. there is no way to know to which species belogns this like because they were inside the speciesinfo likings so it is enstablished that this like belongs to a precise species. in the example rabbit example: if this liking is referred to a rabbit the liked_spec_id is the id of a carrot and the like_factor is how much from 1 to 100 the rabbit likes the carrot

See also

SpeciesInfo
SpeciesInfo::likings

Definition at line 43 of file like.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Like::Like (unsigned int *u_spec_id* = 0, int *u_like_factor* = 0)

default constructor

Parameters

u_spec_id	the species id
u_like_-factor	the like factor

Definition at line 25 of file like.cpp.

6.18.3 Member Function Documentation

6.18.3.1 bool Like::is_full ( ) [virtual]

is the like full

controll if the liked_spec_id is > 0

Implements eco::Container.

Definition at line 35 of file like.cpp.

6.18.4 Member Data Documentation

6.18.4.1 int Like::like_factor

how much the species is liked

positive is attraction

negative is repulsion

0 is 0

Definition at line 85 of file like.h.

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

- sources/like.h
- sources/like.cpp

6.19 LikeFactorCmp Struct Reference

Public Member Functions

- bool **operator()** (const Like &lhs, const Like &rhs)

6.19.1 Detailed Description

Definition at line 45 of file classcompares.hpp.

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

- sources/classcompares.hpp

6.20 LikeRefCmp Struct Reference

used in SpeciesInfo.h

```
#include <classcompares.hpp>
```

Public Member Functions

- bool operator() (const unsigned int &lhs, const unsigned int &rhs)

  returns lhs>rhs

6.20.1 Detailed Description

used in SpeciesInfo.h compares the like_factors of 2 like giving rhe biggerone. this is done to have the highest Like first.

See also

SpeciesInfo::m_likings_by_lk_factor

Definition at line 36 of file classcompares.hpp.

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

- sources/classcompares.hpp

6.21 PopulationVariation Struct Reference

variation of population for a species in a subsystemcontainer

```
#include <populationvariation.hpp>
```

Public Types

- typedef std::pair< unsigned int, unsigned int > coord_tp

  coordinates first x second y

Public Member Functions

- PopulationVariation (coord_tp u_coord_tp, unsigned int u_species_id, int u_variation, long int u_abs_time, float u_rel_time)

  default contstructor

- ~PopulationVariation ()

  deafult bestructor

Public Attributes

- coord_tp subs_coord

  subsystem coordinate in which occours the variation

- unsigned int species_id

  the species of the animal that variate

- int variation

  the variation

- long int abs_time

  absolute time of the variation

- float rel_time

  relative time of the variation

6.21.1 Detailed Description

variation of population for a species in a subsystemcontainer this class contains the population variation for a species in a determinate subsystem container this class is layered inside StepLog class and parsed by EcosystemContainer::step;

See also

StepLog
EcosystemContainer::step
SubsystemContainer
subs_coord
species_id
vartiation

Definition at line 38 of file populationvariation.hpp.

6.21.2 Member Data Documentation

6.21.2.1 int PopulationVariation::variation

the variation

example: if the animal die variation is -1

Definition at line 71 of file populationvariation.hpp.

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

- sources/populationvariation.hpp

6.22 SubsystemContainer::reproduction Struct Reference

boost multyindex::ordered_index tag

```
#include <subsystemcontainer.h>
```

6.22.1 Detailed Description

boost multyindex::ordered_index tag

Definition at line 85 of file subsystemcontainer.h.

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

- sources/subsystemcontainer.h

6.23 SubsystemContainer::spec_id Struct Reference

boost multyindex::ordered_index tag

```
#include <subsystemcontainer.h>
```

6.23.1 Detailed Description

boost multyindex::ordered_index tag

Definition at line 87 of file subsystemcontainer.h.

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

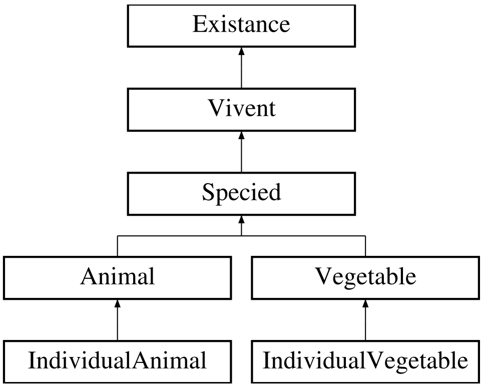
- sources/subsystemcontainer.h

6.24 Specied Class Reference

class Specied the form of life as species belonger

`#include <specied.h>`

Inheritance diagram for Specied:



Public Member Functions

- Specied (unsigned int u_species_id=0, std::string u_species_name="no_name", unsigned int u_life_coast=1, unsigned int u_health_status=50, unsigned int u_calories=1, float u_life_space=10)

    default constructor

- ~Specied ()

    default destructor

- virtual bool is_alive ()

    is the Specied form of life alive?

- unsigned int & species_id ()

    species id

- std::string & species_name ()

    species name

- unsigned int & life_coast ()

    life coast

- unsigned int & health_status ()

    health statis

- unsigned int & calorie ()

calorie

- float & life_space ()

  life space

- unsigned int species_id () const

  get the species_id

- std::string species_name () const

  get the species_name

- unsigned int life_coast () const

  get the life_coast

- unsigned int health_status () const

  get the health_status

- unsigned int calorie () const

  get the calorie

- float life_space () const

  get the life_space

Private Attributes

- unsigned int m_species_id

  the numerical id of the species

- std::string m_species_name

  the name of the species

- unsigned int m_life_coast

  the coast of life

- unsigned int m_health_status

  health status determine when an animal feels good

- unsigned int m_calorie

  calorie the nutritive power of the form of life

- float m_life_space

  occuped space in a quadro

6.24.1 Detailed Description

class Specied the form of life as species belonger probabli the most important class of this library. it rappresents the form of life seen as belonging to a species so it contain all the characteristcs of animal of the same species like the : life coast and ...

Definition at line 41 of file specied.h.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Specied::Specied (unsigned int *u_species_id* = `0`, std::string *u_species_name* = `"no_name"`, unsigned int *u_life_coast* = `1`, unsigned int *u_health_status* = `50`, unsigned int *u_calories* = `1`, float *u_life_space* = `10`)

default constructor

set private data members

Definition at line 34 of file specied.cpp.

6.24.2.2 Specied::∼Specied ( )

default destructor

does nothing

Definition at line 51 of file specied.cpp.

6.24.3 Member Function Documentation

6.24.3.1 unsigned int & Specied::calorie ( )

calorie

See also

m_calorie

Definition at line 85 of file specied.cpp.

6.24.3.2 unsigned int & Specied::health_status ( )

health statis

See also

m_health_status

Definition at line 80 of file specied.cpp.

6.24.3.3 bool Specied::is_alive ( ) [virtual]

is the Specied form of life alive?

returns true if hp() is >= 0; if the animal is not alive it will be removed.

See also

EcosystemContainer::step()

Implements Vivent.

Reimplemented in IndividualAnimal.

Definition at line 60 of file specied.cpp.

6.24.3.4 unsigned int & Specied::life_coast ( )

life coast

See also

m_life_coast

Definition at line 75 of file specied.cpp.

6.24.3.5 float & Specied::life_space ( )

life space

See also

m_life_space

Definition at line 90 of file specied.cpp.

6.24.3.6 unsigned int & Specied::species_id ( )

species id

See also

m_species_id

Definition at line 65 of file specied.cpp.

6.24.3.7 std::string & Specied::species_name ( )

species name

See also

m_species_name

Definition at line 70 of file specied.cpp.

6.24.4 Member Data Documentation

6.24.4.1 unsigned int Specied::m_calorie [private]

calorie the nutritive power of the form of life

each form of life, when eated, aliments the eater which can be only an animal (no carnivorous plants are modelled). the hp the eater receive are hp_eaten ∗ calorie;

Definition at line 161 of file specied.h.

6.24.4.2 unsigned int Specied::m_health_status [private]

health status determine when an animal feels good

the health status has to be read as a percentage of the total hp reachable. over this percentage the form of life starts to feels good, so his libido rise in order to prefer the reproduction. no plant reproduction is included in this model! but for the future realises this data member in included in Specied class

Definition at line 154 of file specied.h.

6.24.4.3 unsigned int Specied::m_life_coast [private]

the coast of life

every time a specied is called it had to pay a life coast. this life coast is sottraed from the m_hp when the life() member is called

See also

live

Definition at line 143 of file specied.h.

6.24.4.4 float Specied::m_life_space [private]

occuped space in a quadro

the space occuped in a quadro in percentage. example: if m_life_space is 10 -> no more than ten animal of this species can be hosted in a quadro

Definition at line 168 of file specied.h.

6.24.4.5 unsigned int Specied::m_species_id [private]

the numerical id of the species

this is the numerical id of the species and is used to identify the species in order to compute parameters and make statistics

Definition at line 125 of file specied.h.

6.24.4.6 std::string Specied::m_species_name [private]

the name of the species

the name of the species as it could be for a human. example: lion, bear, rabbit... this name is NOT used in any computational process, it's only for a better human understanding of the process

See also

m_species_id

Definition at line 135 of file specied.h.

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

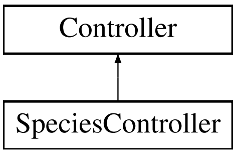
- sources/specied.h
- sources/specied.cpp

6.25 SpeciesController Class Reference

contain info about the species in the ecosystem

`#include <speciescontroller.h>`

Inheritance diagram for SpeciesController:



Public Types

- typedef std::map< unsigned int, SpeciesInfo > infos_tp

  the type of the infos map

- typedef std::map< unsigned int, SpeciesInfo >::iterator infos_it

  info iterator

- typedef std::map< unsigned int, SpeciesInfo >::const_iterator infos_const_it

  info const_iterator

Public Member Functions

- SpeciesController (unsigned int u_species_number=0)

  default contructor

- ∼SpeciesController ()

  default destrictor

- virtual bool check ()

  check if the controll is completed

- std::pair< infos_it, bool > insert (const SpeciesInfo &spi)

  insert a species info

- infos_const_it get_info (const unsigned int u_spec_id)

  get infos about a species

- const SpeciesInfo & get_info_ref (const unsigned int u_spec_id) const

  get the reference to the species info whith u_spec_id as species_id

- int get_like_factor (const unsigned int predator_spec_id, const unsigned int prey_spec_id)

  get like_factor

- const std::map< unsigned int, SpeciesInfo > & get_infos () const

  get the species infos

- std::map< unsigned int, SpeciesInfo > & get_infos ()

  get the infos

- unsigned int species_number ()

  get the total species number

Private Member Functions

- bool string_parser (const std::string &str, SpeciesInfo &spi)

  string parser

Private Attributes

- unsigned int m_species_number

  total number of species in the ecosystem

- std::map< unsigned int, SpeciesInfo > m_infos

  container of species info

Friends

- std::ifstream & operator>> (std::ifstream &is, SpeciesController &sa)

  operator >>

- std::istream & operator>> (std::istream &is, SpeciesController &sa)

  operator >>

- std::ostream & operator<< (std::ostream &is, const SpeciesController &sa)

  operator <<

6.25.1 Detailed Description

contain info about the species in the ecosystem this class contains all the information about the species present in the ecosystem. this class is the UNIQUE reference for the features of a species. this is class inside the ecosystem to provide the values relative the species. in this way al the form of life in the ecosystem of the same species have the same value for dm relative to the species.

this class is also used to determine how much species like each others. for like we intend "to prefer for dinner".

particular attention was given to the affidability and to the rightness of the data field inserted. controll statement preservers from multiple insertions of lines and wrong likes. the computational coast is hight but this element has to be build rightly and only one time for ecosystem so is a good fee to pay.

Definition at line 61 of file speciescontroller.h.

6.25.2 Member Typedef Documentation

6.25.2.1 typedef std::map<unsigned int, SpeciesInfo>::const_iterator SpeciesController::infos_const_it

info const_iterator

See also

m_infos

Definition at line 81 of file speciescontroller.h.

6.25.2.2 typedef std::map<unsigned int, SpeciesInfo>::iterator SpeciesController::infos_it

info iterator

See also

m_infos

Definition at line 75 of file speciescontroller.h.

6.25.2.3 typedef std::map<unsigned int, SpeciesInfo> SpeciesController::infos_tp

the type of the infos map

typedef std::map<unsigned int, SpeciesInfo>

Definition at line 69 of file speciescontroller.h.

6.25.3 Constructor & Destructor Documentation

6.25.3.1 SpecCon::SpeciesController (unsigned int *u_species_number* = 0)

default contructor

if no u_species_number is passed will set to 0 and check() will fail

Definition at line 40 of file speciescontroller.cpp.

6.25.4 Member Function Documentation

6.25.4.1 bool SpecCon::check ( ) [virtual]

check if the controll is completed

Returns

true only if those condition are simultaneus:

- total species number is not 0
- the number of SpeciesInfos coincide with the number of species in ecosystem
- every SpeciesInfo returns true if is_full() is called

See also

SpeciesInfo::is_full()

Implements Controller.

Definition at line 49 of file speciescontroller.cpp.

6.25.4.2 SpecCon::infos_const_it SpecCon::get_info (const unsigned int *u_spec_id*)

get infos about a species

Returns

a const iterator to a pair which the second member is the SpeciesInfo. if the species does not exists returns an iterator to m_infos.end();

Definition at line 178 of file speciescontroller.cpp.

6.25.4.3 const std::map< unsigned int, SpeciesInfo > & SpecCon::get_infos ( ) const

get the species infos

Returns

a const reference to species infos container, which is a map;

Definition at line 225 of file speciescontroller.cpp.

6.25.4.4 int SpecCon::get_like_factor (const unsigned int *predator_spec_id,* const unsigned int *prey_spec_id*)

get like_factor

Parameters

predator_spec_id	the species id of the Species which wants to eat.
prey_spec_id	the species id that will be eaten

Returns

the like factor. how much the predator like the prey

See also

Like::like_factor

Definition at line 202 of file speciescontroller.cpp.

6.25.4.5 bool SpecCon::string_parser (const std::string & *str,* SpeciesInfo & *spi*) [private]

string parser

parse an input line and put's the input in the spi field

Input Format: the line passed contains numerous info divided by a "|" here's the order of the fields: (note that we are in C++ so the first element has position 0)

- species_id (0 is not valid)
- species_name
- "vegetable" or "animal"
- life_coast
- health_status
- calories
- life_space

after 6 tokens have to insert the Like of the species in the format: liked_species-like_-factor. pay attention to the "-". as usual every Like has to be separated by "|". return m_infos; if the numbers of Likes is not = to the total species number in the ecosystem - 1 . the function will return false;

example: if we have 3 species in the ecosystem (1 is lion, 2 is gazzella , 3 is grass):

"1|lion|animal|10|50|1|45|2-50|3-0"

Parameters

str	string containing the data
spi	SpeciesInfo in which put data

Returns

true if the string were well done, false if the string wasn't formatted correctly

See also

operator>>

Definition at line 235 of file speciescontroller.cpp.

6.25.5 Friends And Related Function Documentation

6.25.5.1 std::ifstream& operator>> (std::ifstream & *is*, SpeciesController & *sa*) [friend]

operator >>

load all SpeciesInfo for all species. take the first line of the stream and passes it a string_parser. to be used whith filestream

See also

string_parser for the data format;

Definition at line 571 of file speciescontroller.cpp.

6.25.5.2 std::istream& operator>> (std::istream & *is,* SpeciesController & *sa*) [friend]

operator >>

load all SpeciesInfo for all species. to be used whith std::cin, ask for the fields an compose a line to pass to line_parser then if the line is well composed insert the SpeciesInfo

Definition at line 654 of file speciescontroller.cpp.

6.25.6 Member Data Documentation

6.25.6.1 std::map<unsigned int, SpeciesInfo> SpeciesController::m_infos [private]

container of species info

species infos were sorted by spec_id

See also

SpeciesInfo
SpeciesInfo::operator<

Definition at line 214 of file speciescontroller.h.

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

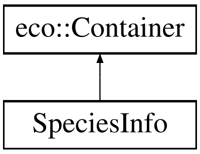
- sources/speciescontroller.h
- sources/speciescontroller.cpp

6.26 SpeciesInfo Struct Reference

species info containers

`#include <speciesinfo.h>`

Inheritance diagram for SpeciesInfo:



Public Types

- typedef std::map< unsigned int, Like > likings_map_tp

  likings type

- typedef likings_map_tp::iterator likings_it

  iteratore per membro di likings

- typedef likings_map_tp::const_iterator likings_const_it

  iteratore constante per membro di likings

- typedef std::multiset< Like, LikeFactorCmp > likings_by_lk_factor_tp

  multimap sorted by like_factor

- typedef likings_by_lk_factor_tp::iterator likings_by_lk_factor_iterator

  iterator to access like sorted by lk factor

- typedef likings_by_lk_factor_tp::const_iterator likings_by_lk_factor_const_iterator

  const iterator to access like sorted by lk factor

Public Member Functions

- SpeciesInfo (unsigned int u_species_id=0, std::string u_species_name="no_name", unsigned int u_life_coast=1, unsigned int u_health_status=50, unsigned int u_calories=1, float u_life_space=10, bool u_is_animal=0, std::map< unsigned int, Like > u_likings=std::map< unsigned int, Like >(), unsigned int u_tot_spec_num=0)

  default constructor

- ∼SpeciesInfo ()

  default destructor

- bool is_full ()

  controll if there is a species id instantiated

- bool insert_like (const Like lk)

  insert a Like in likings map

- bool insert_like (const int u_spec_id, const int u_like_factor)

  create and insert a like whith passed parameter

- int get_like_factor (const unsigned int u_spec_id)

  get the like factor of a liked specied

- bool operator< (const SpeciesInfo &info)

  operator <

- bool operator> (const SpeciesInfo &info)

  operator >

- unsigned int & total_species_number ()

  set total_species_number

- std::string get_info_string ()

  get a string of all the data formatted

Public Attributes

- unsigned int species_id

  the numerical id of the species

- std::string species_name

  the name of the species

- unsigned int life_coast

  the coast of life

- unsigned int health_status

  health status determine when an animal feels good

- unsigned int calorie

  calorie the nutritive power of the form of life

- float life_space

  occuped space in a quadro

- bool is_animal

  true if is animal , false is vegetable

- likings_map_tp likings

  the likings

- likings_by_lk_factor_tp likings_by_lk_factor

  likings sorted by lk_factor

Private Member Functions

- bool check_likings ()

  check if the size of likings is = to m_total_species_number

Private Attributes

- unsigned int m_total_species_number

  total number of species present int the ecosystem

Friends

- std::istream & operator>> (std::istream &is, SpeciesInfo &info)

  operator >>

- std::ostream & operator<< (std::ostream &os, const SpeciesInfo &info)

  operator <<

6.26.1 Detailed Description

species info containers this file only contains a struct layered inside the SpeciesAnalizer class.

this struct contains data relatives to the characteristcs of a species. his centrall role is to give a unique reference for the features of a species. so that different animals of the same species must have the same id

See also

SpeciesAnalizer

Definition at line 53 of file speciesinfo.h.

6.26.2 Member Typedef Documentation

6.26.2.1 typedef std::multiset< Like , LikeFactorCmp > SpeciesInfo::likings_by_lk_factor_tp

multimap sorted by like_factor

key is the like_factor and value is an iterator to the corresponding element in likings_-map

Definition at line 71 of file speciesinfo.h.

6.26.3 Member Function Documentation

6.26.3.1 std::string SpeciesInfo::get_info_string ( )

get a string of all the data formatted

the string is composed as is parsed by SpeciesController::string_parser

See also

SpeciesController

Definition at line 298 of file speciesinfo.cpp.

6.26.3.2 int SpeciesInfo::get_like_factor (const unsigned int *u_spec_id*)

get the like factor of a liked specied

Parameters

u_spec_id	the species id of the liked species

Definition at line 187 of file speciesinfo.cpp.

6.26.3.3 bool SpeciesInfo::insert_like (const int *u_spec_id*, const int *u_like_factor*)

create and insert a like whith passed parameter

See also

Like

Definition at line 144 of file speciesinfo.cpp.

6.26.4 Friends And Related Function Documentation

6.26.4.1 std::ostream& operator<< (std::ostream & *os*, const SpeciesInfo & *info*) [friend]

operator <<

prints al the information stored and the likes

Definition at line 237 of file speciesinfo.cpp.

6.26.4.2 std::istream& operator>> (std::istream & *is*, SpeciesInfo & *info*) [friend]

operator >>

DO NOT USE ME!

Definition at line 220 of file speciesinfo.cpp.

6.26.5 Member Data Documentation

6.26.5.1 unsigned int SpeciesInfo::calorie

calorie the nutritive power of the form of life

See also

Species::m_calorie

Definition at line 172 of file speciesinfo.h.

6.26.5.2 unsigned int SpeciesInfo::health_status

health status determine when an animal feels good

the health status has to be read as a percentage of the total hp reachable. over this percentage the form of life starts to feels good, so his libido rise in order to prefer the reproduction. no plant reproduction is included in this model! but for the future realises this data member in included in Specied class

Definition at line 167 of file speciesinfo.h.

6.26.5.3 bool SpeciesInfo::is_animal

true if is animal , false is vegetable

variable used only for a better understanding of the reader.

Definition at line 183 of file speciesinfo.h.

6.26.5.4 unsigned int SpeciesInfo::life_coast

the coast of life

every time a specied is called it had to pay a life coast. this life coast is sottraed from the m_hp when the life() member is called

See also

live

Definition at line 156 of file speciesinfo.h.

6.26.5.5 float SpeciesInfo::life_space

occuped space in a quadro

the space occuped in a quadro in percentage.

Definition at line 177 of file speciesinfo.h.

6.26.5.6 likings_map_tp SpeciesInfo::likings

the likings

how much a species like others.

the key is the spec_id searched. and the sort is provided by Like::operator< using the default set constructor

the nature of the container ensures the inexistance of two egual species

Todo

> scrivi delle considerazioni finali sul fatto che i multi_index sono più comodi in questi casi anche per emulare una map isi isi

Definition at line 199 of file speciesinfo.h.

6.26.5.7 likings_by_lk_factor_tp SpeciesInfo::likings_by_lk_factor

likings sorted by lk_factor

why i didn't use a smart_ptr map? because i would have had to make the likings_map_tp made of boost smart pointers and then create this multimap. it was too late so i decided to make it composed of iterators instead of pointers, references or copies.

so why you did not use a boost:multiindex container? bacause this project has a didactic scope so i want to get both the experiences in order to have an idea of good and evil of stl vs multiindex.

if your last question is: "is it better to use multiindex in this situation? isn't it?"

the answer is one and only : "YES!"

Definition at line 220 of file speciesinfo.h.

6.26.5.8 unsigned int SpeciesInfo::m_total_species_number [private]

total number of species present int the ecosystem

this member is used to controll if the insertion has been completed

Definition at line 244 of file speciesinfo.h.

6.26.5.9 unsigned int SpeciesInfo::species_id

the numerical id of the species

this is the numerical id of the species and is used to identify the species in order to compute parameters and make statistics

Definition at line 138 of file speciesinfo.h.

6.26.5.10 std::string SpeciesInfo::species_name

the name of the species

the name of the species as it could be for a human. example: lion, bear, rabbit... this name is NOT used in any computational process, it's only for a better human understanding of the process

See also

species_id

Definition at line 148 of file speciesinfo.h.

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

- sources/speciesinfo.h
- sources/speciesinfo.cpp

6.27 StepLog Struct Reference

log of the function step

```
#include <steplog.hpp>
```

Public Member Functions

- StepLog ()

  default contructor

- ∼StepLog ()

  default destructor

Public Attributes

- std::vector< PopulationVariation > variations

  population variation

6.27.1 Detailed Description

log of the function step this class is a wrapper containg population variations. this class is a parameter of Ecosystem::step. each variation is parsed by step function and modify the rappresentation of the ecosystem.

in future realises this class could be used to produce efficiently stats of population trend.

See also

PopulationVariation
EcosystemContainer::step

Definition at line 41 of file steplog.hpp.

6.27.2 Member Data Documentation

6.27.2.1 std::vector<PopulationVariation> StepLog::variations

population variation

See also

PopulationVatiation

Definition at line 56 of file steplog.hpp.

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

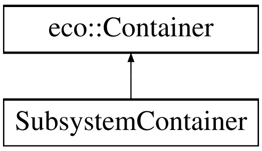
- sources/steplog.hpp

6.28 SubsystemContainer Class Reference

sub ecosystem container

```
#include <subsystemcontainer.h>
```

Inheritance diagram for SubsystemContainer:



Classes

- struct eat

boost multyindex::ordered_index tag

- struct id

  boost multyindex::ordered_index tag

- struct reproduction

  boost multyindex::ordered_index tag

- struct spec_id

  boost multyindex::ordered_index tag

Public Types

- typedef multi_index_container< IndividualAnimal, indexed_by< ordered_non_unique< tag< eat >, composite_key< IndividualAnimal, const_mem_fun< Specied, unsigned int,&IndividualAnimal::species_id >, const_mem_fun< Vivent, unsigned int,&IndividualAnimal::hp > > >, ordered_non_unique< tag< reproduction >, composite_key< IndividualAnimal, const_mem_fun< Specied, unsigned int,&IndividualAnimal::species_id >, const_mem_fun< Vivent, Gender,&IndividualAnimal::gender >, const_mem_fun< Vivent, unsigned int,&IndividualAnimal::hp > > >, ordered_unique< tag< id >, const_mem_fun< Existance, Existance::id_type,&IndividualAnimal::id_number > >, ordered_non_unique< tag< spec_id >, const_mem_fun< Specied, unsigned int,&IndividualAnimal::species_id > > >> animal_set

  animals container typedef

- typedef animal_set::index< eat >::type index_an_by_eat

  typedef for animal index 0 eat

- typedef animal_set::index< reproduction >::type index_an_by_reproduce

  typedef for animal index 1 reproduce

- typedef animal_set::index< id >::type index_an_by_id

  typedef for animal index 2 id_number

- typedef animal_set::index< spec_id >::type index_an_by_spec_id

  typedef for animal index 3 species_id

- typedef index_an_by_eat::iterator an_eat_it

  typedef for eat animal index iterator

- typedef index_an_by_eat::const_iterator an_eat_const_it

  typedef for eat animal index const iterator

- typedef index_an_by_reproduce::iterator an_reproduce_it

  typedef for reproduce animal index iterator

- typedef index_an_by_reproduce::const_iterator an_reproduce_const_it

  typedef for reproduce animal index const iterator

- typedef index_an_by_id::iterator an_id_it

  typedef for id animal index iterator

- typedef index_an_by_id::const_iterator an_id_const_it

  typedef for id animal index const iterator

- typedef index_an_by_spec_id::iterator an_spec_id_it

  typedef for speces id animal index iterator

- typedef index_an_by_spec_id::const_iterator an_spec_id_const_it

  typedef for speces id animal index const iterator

- typedef multi_index_container< IndividualVegetable, indexed_by< ordered_non_unique< tag< eat >, composite_key< IndividualVegetable, const_mem_fun< Specied, unsigned int,&IndividualAnimal::species_id >, const_mem_fun< Vivent, unsigned int,&IndividualAnimal::hp > > >, ordered_unique< tag< id >, const_mem_fun< Existance, Existance::id_type,&IndividualAnimal::id_number > >, ordered_non_unique< tag< spec_id >, const_mem_fun< Specied, unsigned int,&IndividualAnimal::species_id > > >> vegetable_set

  vegetables container typedef

- typedef vegetable_set::index< eat >::type index_veg_by_eat

  typedef for vegetable index 0 eat

- typedef vegetable_set::index< id >::type index_veg_by_id

  typedef for vegetable index 1 id_number

- typedef vegetable_set::index< spec_id >::type index_veg_by_spec_id

  typedef for vegetable index 2 species_id

- typedef index_veg_by_eat::iterator veg_eat_it

  typedef for eat vegetal index iterator

- typedef index_veg_by_eat::const_iterator veg_eat_const_it

  typedef for eat vegetal index const iterator

- typedef index_veg_by_id::iterator veg_id_it

  typedef for id vegetal index iterator

- typedef index_veg_by_id::const_iterator veg_id_const_it

  typedef for id vegetal index const iterator

- typedef index_veg_by_spec_id::iterator veg_spec_id_it *typedef for speces id vegetal index iterator*

- typedef index_veg_by_spec_id::const_iterator veg_spec_id_const_it

  typedef for speces id vegetal index const iterator

- typedef std::pair< animal_set, vegetable_set > subsystem_tp

  the type of the subsystem

Public Member Functions

- SubsystemContainer (unsigned int u_x=0, unsigned int u_y=0)

  default constructor

- ∼SubsystemContainer ()

  default destructor

- virtual bool is_full ()

  is the container full

- virtual bool is_full (Specied &sample)

  is full for this species

- virtual bool is_full (const unsigned int u_spec_id)

  is full for this species

- bool insert (IndividualAnimal &an)

  insert a vivent

- bool insert (IndividualVegetable &veg)

  insert a vivent

- bool remove (const long unsigned int u_id)

  remove an animal

- an_id_it find_animal (const long unsigned int u_id)

  find animal

- veg_id_it find_vegetable (const long unsigned int u_id)

  find vegetable

- std::pair< unsigned int, bool > count_vivents (const unsigned int u_spec_id=0, SpeciesController u_spec_con=SpeciesController(0))

  count the number of vivent in this subsystem

- std::pair< unsigned int, bool > count_vivents (const SpeciesInfo u_spec_info) *count the number of vivent in this subsystem*

- subsystem_tp & sub_ecosystem ()

  set the sub ecosystem

- animal_set & animal_sub_ecosystem ()

  set animal_set

- vegetable_set & vegetable_sub_ecosystem ()

  set vegetable_set

- unsigned int & x_position ()

  set x_position

- unsigned int & y_position ()

  set y_position

- const subsystem_tp & sub_ecosystem () const

  get the sub ecosystem

- const animal_set & animal_sub_ecosystem () const

  get animal_set

- const vegetable_set & vegetable_sub_ecosystem () const

  get vegetable_set

- unsigned int x_position () const

  get m_x_position

- unsigned int y_position () const

  get m_y_position

Private Attributes

- subsystem_tp m_sub_ecosystem

  the sub ecosistem

- unsigned int m_x_position

  x position of the subsystem in the ecosystem

- unsigned int m_y_position

  y position of the subsystem in the ecosystem

Friends

- std::ostream & operator<< (std::ostream &os, const SubsystemContainer &subc)

  ostream operator of SubsystemContainer

6.28.1 Detailed Description

sub ecosystem container this class contain the sub ecosistem composed by all individual animals and vegetables.

Definition at line 69 of file subsystemcontainer.h.

6.28.2 Member Typedef Documentation

6.28.2.1 typedef multi_index_container< IndividualAnimal, indexed_by< ordered_non_unique< tag<eat>, composite_key< IndividualAnimal, const_mem_fun< Specied, unsigned int, &IndividualAnimal::species_id >, const_mem_fun< Vivent, unsigned int, &IndividualAnimal::hp > > >, ordered_non_unique< tag<reproduction>, composite_key< IndividualAnimal, const_mem_fun< Specied, unsigned int, &IndividualAnimal::species_id >, const_mem_fun< Vivent, Gender, &IndividualAnimal::gender >, const_mem_fun< Vivent, unsigned int, &IndividualAnimal::hp > > >, ordered_unique< tag<id>, const_mem_fun< Existance, Existance::id_type , &IndividualAnimal::id_number > >, ordered_non_unique< tag<spec_id>, const_mem_fun< Specied, unsigned int, &IndividualAnimal::species_id > > >> SubsystemContainer::animal_set

animals container typedef

following the boost::multi_index tradition the typedef of this container is particularli cumbersome, but this is a small rate to pay in front of the extreme power of these containers

boost multi_index containers gave us the possibility to sort and index the elements of a single container in different ways.

in addiction we uses the feature of composite_key indexing. this give us the possibility (for example) to have contiguity for all the animals belonging to the same species and then have it sorted by ascendent hp.

in this implementation we have numerous indexes:

- 0 composite key index rappresenting the attitude to be eaten. IndividualAnimal were sorted first by species_id , then by hp.
- 1 composite key index rappresenting the "sexual charming" IndividualAnimal were sorted first by species_id , then by Gender , and finally by hp;
- 2 order unique by the IndividualAnimal id_number. this means that no animals whith same id were admitted

- 3 order non unique by species id, this order is used to fast compute the number of animals of the same species and similar purposes

Definition at line 189 of file subsystemcontainer.h.

6.28.2.2 typedef std::pair<animal_set , vegetable_set> SubsystemContainer::subsystem_tp

the type of the subsystem

first member is an animal_set second is a vegetable_set

Definition at line 334 of file subsystemcontainer.h.

6.28.2.3 typedef multi_index_container< IndividualVegetable, indexed_by< ordered_non_unique< tag<eat>, composite_key< IndividualVegetable, const_mem_fun< Specied, unsigned int, &IndividualAnimal::species_id >, const_mem_fun< Vivent, unsigned int, &IndividualAnimal::hp > > >, ordered_unique< tag<id>, const_mem_fun< Existance, Existance::id_type , &IndividualAnimal::id_number > >, ordered_non_unique< tag<spec_id>, const_mem_fun< Specied, unsigned int, &IndividualAnimal::species_id > > >> SubsystemContainer::vegetable_set

vegetables container typedef

this container is similar but less complicated than the former. this is due to the fact that this version of the project does not implement vegetable reproduction.

Vegetable can only be eaten so the indexes are:

- 0 composite key index rappresenting the attitude to be eaten. IndividualAnimal were sorted first by species_id , then by hp.

- 1 order unique by the IndividualAnimal id_number. this means that no animals whith same id were admitted

- 2 order non unique by species id, this order is used to fast compute the number of animals of the same species and similar purposes

Definition at line 295 of file subsystemcontainer.h.

6.28.3 Constructor & Destructor Documentation

6.28.3.1 SubsystemContainer::SubsystemContainer (unsigned int *u_x* = 0, unsigned int *u_y* = 0)

default constructor

set's the position of the subsystem

Parameters

u_x	x position
u_y	y position

Definition at line 39 of file subsystemcontainer.cpp.

6.28.4 Member Function Documentation

6.28.4.1 std::pair< unsigned int, bool > SubC::count_vivents (const SpeciesInfo *u_spec_info*)

count the number of vivent in this subsystem

returns the number of vivent of a determinate species id if no species id is indicated returns the total ammount of vivent

Parameters

u_spec_info	species info of the species

Returns

a pair whit first member the number of vivent counted, at second member true if the info is correct, false if species info had noot been passed;

Definition at line 426 of file subsystemcontainer.cpp.

6.28.4.2 std::pair< unsigned int, bool > SubC::count_vivents (const unsigned int *u_spec_id* = 0, SpeciesController *u_spec_con* = SpeciesController (0))

count the number of vivent in this subsystem

returns the number of vivent of a determinate species id

Parameters

u_spec_id	the species id of the species
u_spec_con	the species controller. necessary to determinate if vegetable or animal.

Returns

a pair whit first member the number of vivent counted, at second member true if the spec id is correct, false if spec id is not correct or SpeciesController had not been passed

Definition at line 389 of file subsystemcontainer.cpp.

6.28.4.3 SubC::an_id_it SubC::find_animal (const long unsigned int *u_id*)

find animal

returns an iterator to the searched vivent

Parameters

u_id	id of the animal to search to

Returns

id iterator of the id index pointing to the searched animal. if not found returns end()

Definition at line 358 of file subsystemcontainer.cpp.

6.28.4.4 SubC::veg_id_it SubC::find_vegetable (const long unsigned int *u_id*)

find vegetable

returns an iterator to the searched vivent

Parameters

u_id	id of the vegetable to search to

Returns

id iterator of the id index pointing to the searched vegetable. if not found returns end()

Definition at line 372 of file subsystemcontainer.cpp.

6.28.4.5 bool SubC::insert (IndividualAnimal & *an*)

insert a vivent

return true if the vivent had been inserted correctly

Definition at line 284 of file subsystemcontainer.cpp.

6.28.4.6 bool SubC::is_full (Specied & *sample*) [virtual]

is full for this species

Parameters

sample	sample specied animal to control free space

Definition at line 118 of file subsystemcontainer.cpp.

6.28.4.7 bool SubC::is_full ( ) [virtual]

is the container full

potrebbe restare per implementazioni barbare del tipo: c'è ancora spazio per qualcosa?

Implements eco::Container.

Definition at line 278 of file subsystemcontainer.cpp.

6.28.4.8 bool SubC::is_full (const unsigned int *u_spec_id*) [virtual]

is full for this species

Parameters

u_spec_id	is the id of the species to controll

Definition at line 125 of file subsystemcontainer.cpp.

6.28.4.9 bool SubC::remove (const long unsigned int *u_id*)

remove an animal

remove the animal whith specified id return true if the operation succeed

Parameters

u_id	id of the animal to be removed

Definition at line 346 of file subsystemcontainer.cpp.

6.28.5 Friends And Related Function Documentation

6.28.5.1 std::ostream& operator<< (std::ostream & *os*, const SubsystemContainer & *subc*) [friend]

ostream operator of SubsystemContainer

modify the stream printing:

- the susbsytem coordinates
- animal_set size
- vegetable_set size
- all the animals and the vegetables

6.28.6 Member Data Documentation

6.28.6.1 subsystem_tp SubsystemContainer::m_sub_ecosystem [private]

the sub ecosistem

contains animals and vegetables

Definition at line 489 of file subsystemcontainer.h.

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

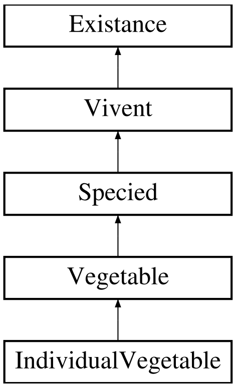
- sources/subsystemcontainer.h
- sources/subsystemcontainer.cpp

6.29 Vegetable Class Reference

class Vegetable

`#include <vegetable.h>`

Inheritance diagram for Vegetable:



Public Member Functions

- Vegetable ()

  deafult constructor

- ~Vegetable ()

  default destructor

6.29.1 Detailed Description

class Vegetable this class is present only for a modeling purpose. his implementation is given to future generations.

Definition at line 36 of file vegetable.h.

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

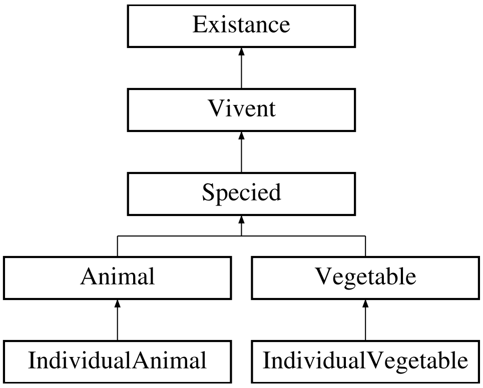
- sources/vegetable.h
- sources/vegetable.cpp

6.30 Vivent Class Reference

class Vivent contain HP

`#include <vivent.h>`

Inheritance diagram for Vivent:



Public Member Functions

- Vivent (unsigned int u_hp=100, Gender u_gender=Gender("asexual"))

  default constructor

- ~Vivent ()

  default destructor

- virtual bool is_alive ()=0

  is_alive

- unsigned int & hp ()

  the hp

- Gender & gender ()

  the gender

- unsigned int hp () const

  get the hp

- Gender gender () const

  get the gender

Private Attributes

- unsigned int m_hp

  HP most important parameter.

- Gender m_gender

  gender of the form of life

6.30.1 Detailed Description

class Vivent contain HP the form of life seen as an object able to live

Definition at line 39 of file vivent.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 Vivent::Vivent (unsigned int *u_hp* = `100`, Gender *u_gender* = Gender`("asexual")`)

default constructor

set's dm, default is hp = 100, gender asexual;

Definition at line 34 of file vivent.cpp.

6.30.2.2 Vivent::∼Vivent ( )

default destructor

does nothing

Definition at line 41 of file vivent.cpp.

6.30.3 Member Function Documentation

6.30.3.1 Gender & Vivent::gender ( )

the gender

See also

m_gender

Definition at line 55 of file vivent.cpp.

6.30.3.2 unsigned int & Vivent::hp ( )

the hp

See also

m_hp

Definition at line 45 of file vivent.cpp.

6.30.3.3 virtual bool Vivent::is_alive ( ) [pure virtual]

is_alive

See also

Existance::is_alive()

Implements Existance.

Implemented in IndividualAnimal, and Specied.

6.30.4 Member Data Documentation

6.30.4.1 Gender Vivent::m_gender [private]

gender of the form of life

every form of life could be male, female, hermaphrodite (both) or asexual (nothing). it's not in our purpose to implement vegetable's gender and hermaphrodite. but, for correctnes and thinking to future realises, gender is included in this adstract class. gender is default setted to no asexual.

Definition at line 91 of file vivent.h.

6.30.4.2 unsigned int Vivent::m_hp [private]

HP most important parameter.

it rappresents the health of the form of life whith a value of 0 the animal is dead whith a value of 100 the animal feels very good

Definition at line 82 of file vivent.h.

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

- sources/vivent.h
- sources/vivent.cpp

Chapter 7

File Documentation

7.1 sources/animal.cpp File Reference

```
#include "animal.h"
```

7.1.1 Detailed Description

implementation of class Animal

Definition in file animal.cpp.

7.2 sources/animal.h File Reference

```
#include "specied.h"
```

Classes

- class Animal

  class Animal

7.2.1 Detailed Description

interface of class Animal

Definition in file animal.h.

7.3 sources/beeings.h File Reference

```
#include "existance.h"
#include "vivent.h"
#include "specied.h"
#include "animal.h"
#include "individualanimal.h"
#include "vegetable.h"
#include "individualvegetable.h"
```

7.3.1 Detailed Description

this file is a wrapper include for al the librarys concerning the form of lifes as:

- Existance
- Vivent
- Specied
- Animal
- Vegetable
- IndividualAnimal
- IndividualVegetable

Definition in file beeings.h.

7.4 sources/classcompares.hpp File Reference

Classes

- struct LikeRefCmp

  used in SpeciesInfo.h

- struct LikeFactorCmp

7.4.1 Detailed Description

file containing the functors used to compare classes

Definition in file classcompares.hpp.

7.5 sources/controller.h File Reference

Classes

- class Controller

  class that generally controll

7.5.1 Detailed Description

this file contains the container abstract class

Definition in file controller.h.

7.6 sources/ecosystem.h File Reference

```
#include <string>
#include <iostream>
#include "boost/multi_array.hpp"
#include <boost/random/mersenne_twister.hpp>
#include <boost/random/uniform_int.hpp>
#include <boost/random/variate_generator.hpp>
#include "container.h"
#include "subsystemcontainer.h"
#include "speciescontroller.h"
#include "steplog.hpp"
```

Classes

- class EcosystemContainer

  contains all the form of life

7.6.1 Detailed Description

the ecosystem contains all the form of life. is divided in subecosystems.

See also

subecosystem

Definition in file ecosystem.h.

7.7 sources/existance.cpp File Reference

```
#include "existance.h"
```

7.7.1 Detailed Description

implementation of the Abstract class Existance

Definition in file existance.cpp.

7.8 sources/existance.h File Reference

```
#include "time.h"
```

Classes

- class Existance

  class Existance the most abstracted object

7.8.1 Detailed Description

interface of the Abstract class Existance

Definition in file existance.h.

7.9 sources/fieldchangers.hpp File Reference

```
#include "beeings.h"
```

Classes

- struct change_animal_hp

  functor to change IndividualAnimal hp

- struct change_animal_libido

  functor to cahnge IndividualAnimal libido

- struct change_animal_appetite

  functor to change IndividualAnimal appetite

7.9.1 Detailed Description

this file include numberous functors (or unary function) used to change fields of individual animals or individual vegetables (passed to the index::modify() member).

Definition in file fieldchangers.hpp.

7.10 sources/gender.cpp File Reference

```
#include <iostream>
#include "gender.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const Gender &gen)

  ostream operator of gender

7.10.1 Detailed Description

class Gender implementation

Definition in file gender.cpp.

7.10.2 Function Documentation

7.10.2.1 std::ostream& operator<< (std::ostream & *os*, const Gender & *gen*)

ostream operator of gender

prints a string saying the actual gender is a wrapper of Gender::gender()

Definition at line 213 of file gender.cpp.

7.11 sources/gender.h File Reference

gender class interface

```
#include <string>
#include <iostream>
```

Classes

- class Gender

the gender of the form of life

7.11.1 Detailed Description

gender class interface

Definition in file gender.h.

7.12 sources/individualanimal.cpp File Reference

```
#include <iostream>
#include "individualanimal.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const IndividualAnimal &an)

7.12.1 Detailed Description

contains the implementation of IndividualAnimal

Definition in file individualanimal.cpp.

7.12.2 Function Documentation

7.12.2.1 std::ostream& operator<< (std::ostream & *os*, const IndividualAnimal & *an*)

prints al main info of the animal

Definition at line 76 of file individualanimal.cpp.

7.13 sources/individualanimal.h File Reference

```
#include "animal.h"
```

Classes

- class IndividualAnimal

  class IndividualAnimal

7.13.1 Detailed Description

this file contains the interface of IndividualAnimal

Definition in file individualanimal.h.

7.14 sources/individualvegetable.h File Reference

```
#include "vegetable.h"
```

Classes

- class IndividualVegetable

  class IndividualVegetable

7.14.1 Detailed Description

implementation of IndividualVegetable

interface of IndividualVegetable

Definition in file individualvegetable.h.

7.15 sources/miscellaneus.h File Reference

wrapper containing miscellaneus and varius classes

```
#include "gender.h"
```

7.15.1 Detailed Description

wrapper containing miscellaneus and varius classes

Definition in file miscellaneus.h.

7.16 sources/specied.h File Reference

```
#include "vivent.h"
#include <string>
```

Classes

- class Specied

class Specied the form of life as species belonger

7.16.1 Detailed Description

implementation of Specied

this file contains the interface of the class Specied. this class is one of the most important.

Definition in file specied.h.

7.17 sources/speciescontroller.h File Reference

```
#include <map>
#include <iostream>
#include <fstream>
#include <string>
#include "controller.h"
#include "speciesinfo.h"
```

Classes

- class SpeciesController

  contain info about the species in the ecosystem

7.17.1 Detailed Description

SpeciesController interface

Definition in file speciescontroller.h.

7.18 sources/speciesinfo.h File Reference

```
#include <string>
#include <map>
#include <iostream>
#include <set>
#include "boost/lexical_cast.hpp"
#include "specied.h"
```

```
#include "like.h"
#include "container.h"
#include "classcompares.hpp"
```

Classes

- struct SpeciesInfo

  species info containers

7.18.1 Detailed Description

Definition in file speciesinfo.h.

7.19 sources/subsystemcontainer.cpp File Reference

```
#include <iostream>
#include "subsystemcontainer.h"
```

Typedefs

- typedef SubsystemContainer SubC

Functions

- std::ostream & operator<< (std::ostream &os, const SubC &subc)

  ostream operator of SubsystemContainer

7.19.1 Detailed Description

Definition in file subsystemcontainer.cpp.

7.19.2 Typedef Documentation

7.19.2.1 typedef SubsystemContainer SubC

implementaions of subsystemcontainer.h

Definition at line 35 of file subsystemcontainer.cpp.

7.19.3 Function Documentation

7.19.3.1 std::ostream& operator<< (std::ostream & *os*, const SubC & *subc*)

ostream operator of SubsystemContainer

modify the stream printing:

- the susbsytem coordinates
- animal_set size
- vegetable_set size
- all the animals and the vegetables

Definition at line 488 of file subsystemcontainer.cpp.

7.20 sources/subsystemcontainer.h File Reference

```
#include <utility>
#include <boost/multi_index_container.hpp>
#include <boost/multi_index/ordered_index.hpp>
#include <boost/multi_index/identity.hpp>
#include <boost/multi_index/member.hpp>
#include <boost/multi_index/mem_fun.hpp>
#include <boost/multi_index/composite_key.hpp>
#include "container.h"
#include "beeings.h"
#include "speciescontroller.h"
#include "speciesinfo.h"
```

Classes

- class SubsystemContainer

  sub ecosystem container

- struct SubsystemContainer::id

  boost multyindex::ordered_index tag

- struct SubsystemContainer::eat

  boost multyindex::ordered_index tag

- struct SubsystemContainer::reproduction

  boost multyindex::ordered_index tag

- struct SubsystemContainer::spec_id

  boost multyindex::ordered_index tag

7.20.1 Detailed Description

file of the subsystem container, it contains the specification of the container.

Definition in file subsystemcontainer.h.

7.21 sources/time.h File Reference

classes to manipulate and determine the time of the system

```
#include <iostream>
```

Classes

- class AbstractClock

  abstract class for the clock

- class Clock

  real clock able to give the time of the sistem

- class DateOfBirth

  simple class for the date of birh

7.21.1 Detailed Description

classes to manipulate and determine the time of the system is really difficult to determine time in this context.

we have developed two types of time: relative and absolute

relative time: the easyest way is to think to what is a ecosistem cicle. An ecosistem cicle is concluded when all the animals in the ecosistem where called. for "call" we intend every time a form of life interact with another form of life, so if an animal fight whith another this constitutes o total of 2 calls) as you can imagine it could be really difficult to controll that all the animals were called so if we give to te ecosistem cicle a value of 1, each call to an animal has the time value interval of 1/total_number_of_forms_of_life present in the ecosistem. so the running relative time is the number of cicles passed and the quantiti of the cicle running absolute time: nothing different from the number of calls occured from the creation of the first form of life

Definition in file time.h.

7.22 sources/vegetable.cpp File Reference

```
#include "vegetable.h"
```

7.22.1 Detailed Description

implementation of class Vegetable

Definition in file vegetable.cpp.

7.23 sources/vegetable.h File Reference

```
#include "specied.h"
```

Classes

- class Vegetable

  class Vegetable

7.23.1 Detailed Description

interface of class Vegetable

Definition in file vegetable.h.

7.24 sources/vivent.h File Reference

```
#include "existance.h"
#include "miscellaneus.h"
```

Classes

- class Vivent

  class Vivent contain HP

7.24.1 Detailed Description

this file contains the implementation of Vivent

this file contains the interface of Vivent abstract class

Definition in file vivent.h.

Index