

Reference Manual

Generated by Doxygen 1.7.1

Fri Apr 8 2011 21:22:14

Contents

1	Gridhunt2: The famous Gridhunt game	1
1.1	Objective	1
1.2	How to Start	1
1.3	Compilation and Running the Game	1
1.4	Copyright	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	Coord Class Reference	9
5.1.1	Detailed Description	9
5.1.2	Constructor & Destructor Documentation	9
5.1.2.1	Coord	9
5.1.3	Member Function Documentation	10
5.1.3.1	distance	10
5.1.4	Friends And Related Function Documentation	10
5.1.4.1	operator<<	10
5.2	Die Class Reference	10
5.2.1	Detailed Description	10
5.2.2	Member Function Documentation	10
5.2.2.1	roll	10
5.3	Grid Class Reference	10
5.3.1	Detailed Description	11

5.3.2	Member Function Documentation	11
5.3.2.1	checkCoord	11
5.3.2.2	getMaxE	11
5.3.2.3	getMaxN	11
5.3.2.4	randomCoord	11
5.3.3	Member Data Documentation	11
5.3.3.1	die	11
5.4	Gridhunt Class Reference	11
5.4.1	Detailed Description	12
5.4.2	Constructor & Destructor Documentation	12
5.4.2.1	Gridhunt	12
5.4.3	Member Function Documentation	12
5.4.3.1	leaderBoard	12
5.4.3.2	run	12
5.4.3.3	show	12
5.5	Hunter Class Reference	12
5.5.1	Detailed Description	13
5.5.2	Member Function Documentation	13
5.5.2.1	distanceToMonster	13
5.5.2.2	makeMove	13
5.5.2.3	positionOfMonster	13
5.5.2.4	setMonster	13
5.6	HunterChecker Class Reference	14
5.6.1	Detailed Description	14
5.6.2	Member Function Documentation	14
5.6.2.1	makeMove	14
5.7	HunterRandom Class Reference	14
5.7.1	Detailed Description	15
5.7.2	Member Function Documentation	15
5.7.2.1	makeMove	15
5.8	Mobile Class Reference	15
5.8.1	Detailed Description	16
5.8.2	Member Enumeration Documentation	16
5.8.2.1	Direction	16
5.8.3	Constructor & Destructor Documentation	16
5.8.3.1	Mobile	16

5.8.4	Member Function Documentation	16
5.8.4.1	getPosition	16
5.8.4.2	getRound	17
5.8.4.3	makeMove	17
5.8.4.4	move	17
5.8.4.5	pay	17
5.8.4.6	say	17
5.8.4.7	teleport	17
5.8.5	Friends And Related Function Documentation	17
5.8.5.1	operator<<	17
5.8.6	Member Data Documentation	17
5.8.6.1	directionName	17
5.9	Monster Class Reference	18
5.9.1	Member Function Documentation	18
5.9.1.1	makeMove	18
6	File Documentation	19
6.1	constants.h File Reference	19
6.1.1	Detailed Description	19
6.1.2	Function Documentation	19
6.1.2.1	ERROR_COORD	19
6.1.3	Variable Documentation	19
6.1.3.1	ERROR_DIST	19

Chapter 1

Gridhunt2: The famous Gridhunt game

Idea by Werner Schönfeldinger and reimplemented by Michael Hahsler

1.1 Objective

The objective of gridhunt is to implement a hunter (i.e., a subclass of [Hunter](#)) who can catch the monster faster than all other hunters. [Gridhunt](#) uses a 30x30 grid of squares. A monster moves on this grid (see class [Monster](#) for how exactly the monster moves). [Gridhunt](#) is turn based. Each turn each hunter gets between 1 and 5 points which she/he can use for actions (e.g., move to an adjacent square, find out where the monster is; teleport to a random location on the grid). A hunter catches the monster if she/he moves on the same square the monster currently occupies. If the monster survives 100 rounds it wins.

1.2 How to Start

Download the code from [the Gridhunt2 web site](#). Make sure you check out the copyright notice below. Start with a copy of [HunterRandom](#), give it a new class name. Then edit `gridhunt_start.cc` to include your new hunter in the hunting party. Modify the new hunter's `makeMove()` function to make her/him smarter.

1.3 Compilation and Running the Game

Add your class to the Makefile (line 29) and type `make` in your terminal. Run the game with `./gridhunt`. Alternatively you can import the project into NetBeans and run it from there.

Good luck!

1.4 Copyright

Copyright (C) 2010 [Michael Hahsler](#)

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<http://www.gnu.org/licenses/>](http://www.gnu.org/licenses/).

Chapter 2

Class Index

2.1 Class Hierarchy

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

Coord	9
Die	10
Grid	10
Gridhunt	11
Mobile	15
Hunter	12
HunterChecker	14
HunterRandom	14
Monster	18

Chapter 3

Class Index

3.1 Class List

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

Coord	9
Die	10
Grid	10
Gridhunt	11
Hunter	12
HunterChecker	14
HunterRandom	14
Mobile	15
Monster	18

Chapter 4

File Index

4.1 File List

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

constants.h	19
Coord.h	??
Die.h	??
Grid.h	??
Gridhunt.h	??
Hunter.h	??
HunterChecker.h	??
HunterRandom.h	??
Mobile.h	??
Monster.h	??

Chapter 5

Class Documentation

5.1 Coord Class Reference

```
#include <Coord.h>
```

Public Member Functions

- `Coord` (int *n*=0, int *e*=0)
- int `getN` () const
- int `getE` () const
- void `setCoord` (int, int)
- int `distance` (const `Coord` &) const
- bool `operator==` (const `Coord` &) const
- bool `operator!=` (const `Coord` &right) const
- `Coord operator+` (const `Coord` &) const

Friends

- ostream & `operator<<` (ostream &, const `Coord` &)

5.1.1 Detailed Description

Coordinates

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `Coord::Coord (int n = 0, int e = 0)`

Create a coordinate object (North/East)

5.1.3 Member Function Documentation

5.1.3.1 `int Coord::distance (const Coord & arg2) const`

Get distance to another [Coord](#).

5.1.4 Friends And Related Function Documentation

5.1.4.1 `ostream& operator<< (ostream & o, const Coord & c) [friend]`

global insertion operator for [Coord](#)

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

- Coord.h
- Coord.cc

5.2 Die Class Reference

```
#include <Die.h>
```

Public Member Functions

- `int roll (int max=6) const`

5.2.1 Detailed Description

simple multi-sided die.

5.2.2 Member Function Documentation

5.2.2.1 `int Die::roll (int max = 6) const`

get a random number between 1 and max.

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

- Die.h
- Die.cc

5.3 Grid Class Reference

```
#include <Grid.h>
```


Public Member Functions

- **Grid** (int, int)
- bool **checkCoord** (const **Coord** &) const
- **Coord** **randomCoord** () const
- int **getMaxN** () const
- int **getMaxE** () const

Public Attributes

- const **Die** **die**

5.3.1 Detailed Description

Some helper functions needed for the grid

5.3.2 Member Function Documentation

5.3.2.1 bool Grid::checkCoord (const Coord & *aCoord*) const

check if some coordinates are inside the grid

5.3.2.2 int Grid::getMaxE () const

get the number of squares along the y-axis. The squares are numbered 0 to maxY-1.

5.3.2.3 int Grid::getMaxN () const

get the number of squares along the x-axis. The squares are numbered 0 to maxX-1.

5.3.2.4 Coord Grid::randomCoord () const

produce a random location

5.3.3 Member Data Documentation

5.3.3.1 const Die Grid::die

the grid has its die

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

- Grid.h
- Grid.cc

5.4 Gridhunt Class Reference

```
#include <Gridhunt.h>
```

Public Member Functions

- [Gridhunt](#) (int maxN=30, int maxE=30, int maxRounds=100)
- void [run](#) (bool visual=true)
- void [show](#) () const
- void [leaderBoard](#) () const
- void [setHunters](#) ([Hunter](#) *[], int)
- void [setMonster](#) ([Monster](#) *)

5.4.1 Detailed Description

Main class of the game.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 `Gridhunt::Gridhunt (int maxN = 30, int maxE = 30, int maxRounds = 100)`

Create a game with a grid (default: 30x30) and run the game for a maximum of maxRounds rounds

5.4.3 Member Function Documentation

5.4.3.1 `void Gridhunt::leaderBoard () const`

Display current leader board

5.4.3.2 `void Gridhunt::run (bool visual = true)`

Run the game. Turn detailed display on/off with visual.

5.4.3.3 `void Gridhunt::show () const`

Display [Mobile](#) objects on [Grid](#)

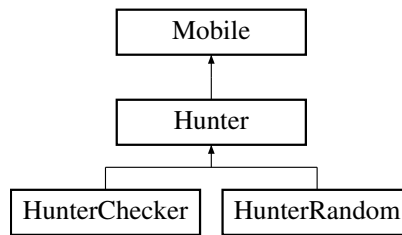
The documentation for this class was generated from the following files:

- Gridhunt.h
- Gridhunt.cc

5.5 Hunter Class Reference

```
#include <Hunter.h>
```

Inheritance diagram for Hunter:



Public Member Functions

- **Hunter** (const string &name="A player", char decal= 'x')
- virtual void [makeMove](#) ()=0
- int [distanceToMonster](#) ()
- [Coord](#) [positionOfMonster](#) ()
- void [setMonster](#) ([Monster](#) *monster)

5.5.1 Detailed Description

A [Hunter](#). Inherit from this class to implement your player class

5.5.2 Member Function Documentation

5.5.2.1 int Hunter::distanceToMonster ()

get the distance to the monster in moves. Cost: 1 point.

5.5.2.2 virtual void Hunter::makeMove () [pure virtual]

Overwrite this function to implement your player's behavior. You can use the public member functions in [Mobile](#) and [Hunter](#) or in other classes. Remember that you only have 1-5 points per round and move, teleport, getPosition, distanceToMonster and positionOfMonster (some of them are inherited from [Mobile](#)) cost points.

Implements [Mobile](#).

Implemented in [HunterChecker](#), and [HunterRandom](#).

5.5.2.3 Coord Hunter::positionOfMonster ()

get the position of the monster. Cost: 2 points.

5.5.2.4 void Hunter::setMonster ([Monster](#) * *monster*) [inline]

this is only used by the game to initialize your "monster scanner" which allows you to use [distanceToMonster\(\)](#) and [positionOfMonster\(\)](#).

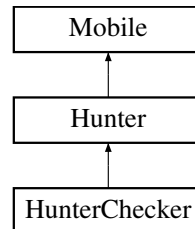
The documentation for this class was generated from the following files:

- Hunter.h
- Hunter.cc

5.6 HunterChecker Class Reference

```
#include <HunterChecker.h>
```

Inheritance diagram for HunterChecker:



Public Member Functions

- **HunterChecker** (const string &name="Checker", char decal= 'x')
- void [makeMove](#) ()

5.6.1 Detailed Description

This player checks constantly where the monster is but is too lazy to go anywhere.

5.6.2 Member Function Documentation

5.6.2.1 void HunterChecker::makeMove () [virtual]

I'm just checking where the monster is.

Implements [Hunter](#).

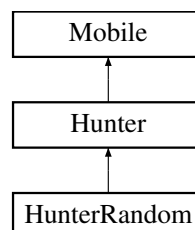
The documentation for this class was generated from the following files:

- HunterChecker.h
- HunterChecker.cc

5.7 HunterRandom Class Reference

```
#include <HunterRandom.h>
```

Inheritance diagram for HunterRandom:



Public Member Functions

- **HunterRandom** (const string &name="Random [Hunter](#)", char decal= 'x')
- void [makeMove](#) ()

5.7.1 Detailed Description

This player randomly moves around and hopes to catch the monster

5.7.2 Member Function Documentation

5.7.2.1 void HunterRandom::makeMove () [virtual]

If I only run around fast enough then I will surely catch the monster!

Implements [Hunter](#).

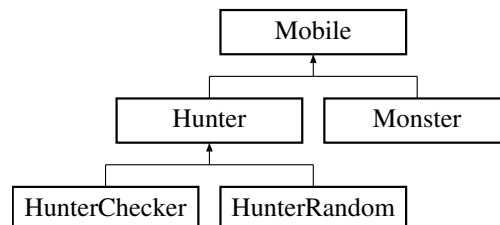
The documentation for this class was generated from the following files:

- HunterRandom.h
- HunterRandom.cc

5.8 Mobile Class Reference

```
#include <Mobile.h>
```

Inheritance diagram for Mobile:



Public Types

- enum [Direction](#) {
 N = 1, NE, E, SE,
 S, SW, W, NW }

Public Member Functions

- [Mobile](#) (const string &name="Some [Mobile](#)", char decal= 'x')
- void [say](#) (const string &) const
- bool [pay](#) (int)
- [Coord](#) [getPosition](#) ()

- bool [teleport](#) ()
- bool [move](#) ([Direction](#))
- virtual void [makeMove](#) ()=0
- [Die](#) * [getDie](#) () const
- string [getName](#) () const
- int [getWins](#) () const
- char [getDecal](#) () const
- int [getPoints](#) () const
- [Grid](#) * [getGrid](#) () const
- int [getRound](#) () const

Static Public Attributes

- static const string [directionName](#) []

Friends

- class [Gridhunt](#)
- class [Hunter](#)
- ostream & [operator<<](#) (ostream &, const [Mobile](#) &)

5.8.1 Detailed Description

Implements a mobile object (either [Hunter](#) or [Monster](#)).

5.8.2 Member Enumeration Documentation

5.8.2.1 enum [Mobile::Direction](#)

enumeration of possible directions.

5.8.3 Constructor & Destructor Documentation

5.8.3.1 [Mobile::Mobile](#) (const string & *name* = "*Some Mobile*", char *decal* = '*x*')

Simple constructor.

Parameters

name Name of the mobile.

decal Decal used for displaying the mobile on a grid.

5.8.4 Member Function Documentation

5.8.4.1 [Coord Mobile::getPosition](#) ()

get your own position. Cost: 1 point.

5.8.4.2 `int Mobile::getRound () const [inline]`

get the round number (rounds start with 1)

5.8.4.3 `virtual void Mobile::makeMove () [pure virtual]`

this member function will be called once per round. It is a pure virtual function and has to be overloaded in your subclass in order to implement the mobile's behavior.

Implemented in [Hunter](#), [HunterChecker](#), [HunterRandom](#), and [Monster](#).

5.8.4.4 `bool Mobile::move (Direction dir)`

move the mobile one square in a given direction. Cost: 1 point.

5.8.4.5 `bool Mobile::pay (int howMuch)`

reduce your available points for this round.

5.8.4.6 `void Mobile::say (const string & message) const`

say something (don't use cout)

5.8.4.7 `bool Mobile::teleport ()`

teleport you to a random location on the grid. Cost: 2 points.

5.8.5 Friends And Related Function Documentation

5.8.5.1 `ostream& operator<< (ostream & o, const Mobile & m) [friend]`

displays mobile name and location

5.8.6 Member Data Documentation

5.8.6.1 `const string Mobile::directionName [static]`

Initial value:

```
{ "X", "N", "NE", "E", "SE",  
  "S", "SW", "W", "NW" }
```

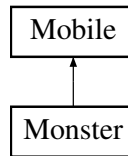
string representaion of directions for printing.

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

- Mobile.h
- Mobile.cc

5.9 Monster Class Reference

Inheritance diagram for Monster:



Public Member Functions

- **Monster** (const string &name="A nameless monster", char decal= 'M')
- void [makeMove](#) ()

5.9.1 Member Function Documentation

5.9.1.1 void Monster::makeMove () [virtual]

this member function will be called once per round. It is a pure virtual function and has to be overloaded in your subclass in order to implement the mobile's behavior.

Implements [Mobile](#).

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

- Monster.h
- Monster.cc

Chapter 6

File Documentation

6.1 constants.h File Reference

```
#include "Coord.h"
```

Functions

- const [Coord ERROR_COORD](#) (-1,-1)

Variables

- const int [ERROR_DIST](#) = -1

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 const Coord ERROR_COORD (- 1, - 1)

error coordinate returned by functions if getting the coordinate failed (e.g., you do not have enough points left).

6.1.3 Variable Documentation

6.1.3.1 const int ERROR_DIST = -1

error distance returned by functions if getting the distance failed (e.g., you do not have enough points left).

Index

checkCoord
 Grid, [11](#)
constants.h, [19](#)
 ERROR_COORD, [19](#)
 ERROR_DIST, [19](#)
Coord, [9](#)
 Coord, [9](#)
 distance, [10](#)
 operator<<, [10](#)

Die, [10](#)
 roll, [10](#)
die
 Grid, [11](#)
Direction
 Mobile, [16](#)
directionName
 Mobile, [17](#)
distance
 Coord, [10](#)
distanceToMonster
 Hunter, [13](#)

ERROR_COORD
 constants.h, [19](#)
ERROR_DIST
 constants.h, [19](#)

getMaxE
 Grid, [11](#)
getMaxN
 Grid, [11](#)
getPosition
 Mobile, [16](#)
getRound
 Mobile, [16](#)
Grid, [10](#)
 checkCoord, [11](#)
 die, [11](#)
 getMaxE, [11](#)
 getMaxN, [11](#)
 randomCoord, [11](#)
Gridhunt, [11](#)
 Gridhunt, [12](#)
 leaderBoard, [12](#)

run, [12](#)
show, [12](#)

Hunter, [12](#)
 distanceToMonster, [13](#)
 makeMove, [13](#)
 positionOfMonster, [13](#)
 setMonster, [13](#)
HunterChecker, [14](#)
 makeMove, [14](#)
HunterRandom, [14](#)
 makeMove, [15](#)

leaderBoard
 Gridhunt, [12](#)

makeMove
 Hunter, [13](#)
 HunterChecker, [14](#)
 HunterRandom, [15](#)
 Mobile, [17](#)
 Monster, [18](#)
Mobile, [15](#)
 Direction, [16](#)
 directionName, [17](#)
 getPosition, [16](#)
 getRound, [16](#)
 makeMove, [17](#)
 Mobile, [16](#)
 move, [17](#)
 operator<<, [17](#)
 pay, [17](#)
 say, [17](#)
 teleport, [17](#)
Monster, [18](#)
 makeMove, [18](#)
move
 Mobile, [17](#)

operator<<
 Coord, [10](#)
 Mobile, [17](#)

pay
 Mobile, [17](#)
positionOfMonster

- Hunter, [13](#)
- randomCoord
 - Grid, [11](#)
- roll
 - Die, [10](#)
- run
 - Gridhunt, [12](#)
- say
 - Mobile, [17](#)
- setMonster
 - Hunter, [13](#)
- show
 - Gridhunt, [12](#)
- teleport
 - Mobile, [17](#)