swi2010

Generated by Doxygen 1.7.2

Wed Nov 24 2010 11:02:23

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

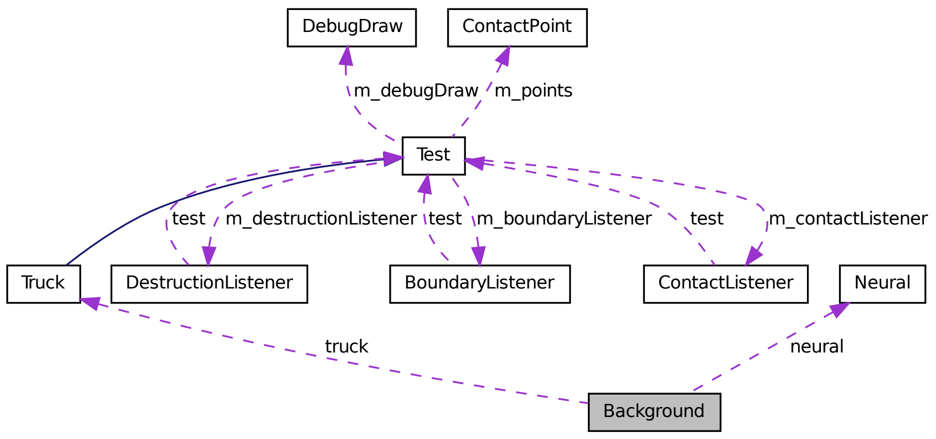
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Background Class Reference

Collaboration diagram for Background:



Public Member Functions

- Background (bool debug)
- Background (double ∗networkWeights, bool debug)
- ∼Background ()

  Destructs the instance.

- double Fitness (Test ∗test)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Background::Background (bool *debug*)

Constructs a new background

Parameters

`in`	*debug*	turn on debugging or not

4.1.1.2 Background::Background (double ∗ *networkWeights,* bool *debug*)

Constructs a background and set network weights

Parameters

`in`	*network-Weights*	Weights used to set the network with
`in`	*debug*	Turn on debugging or not

4.1.2 Member Function Documentation

4.1.2.1 double Background::Fitness (Test ∗ *test*)

Calculates the fitness of the solution (networkWeights) by simulating it

Parameters

`in`	*test*	The environment to simulate in

Returns

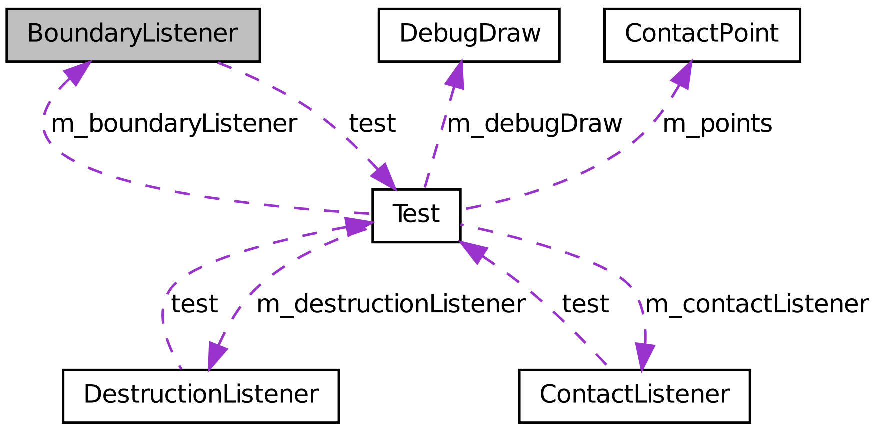
The travelled distance

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

- ai/background.h
- ai/background.cpp

4.2 BoundaryListener Class Reference

Collaboration diagram for BoundaryListener:



Public Member Functions

- void **Violation** (b2Body ∗body)

Public Attributes

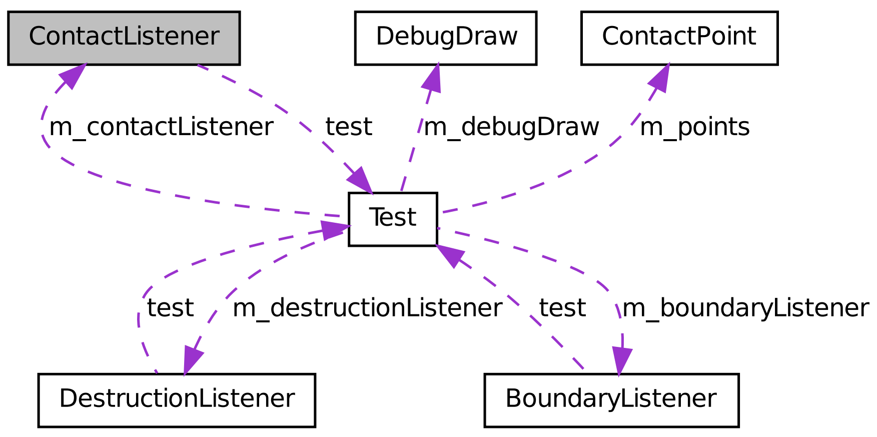
- Test ∗ **test**

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

- Framework/Test.h
- Framework/Test.cpp

4.3 ContactListener Class Reference

Collaboration diagram for ContactListener:



Public Member Functions

- void **Add** (const b2ContactPoint ∗point)
- void **Persist** (const b2ContactPoint ∗point)
- void **Remove** (const b2ContactPoint ∗point)

Public Attributes

- Test ∗ **test**

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

- Framework/Test.h
- Framework/Test.cpp

4.4 ContactPoint Struct Reference

Public Attributes

- b2Shape ∗ **shape1**

- b2Shape ∗ **shape2**
- b2Vec2 **normal**
- b2Vec2 **position**
- b2Vec2 **velocity**
- b2ContactID **id**
- ContactState **state**

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

- Framework/Test.h

4.5 DebugDraw Class Reference

Public Member Functions

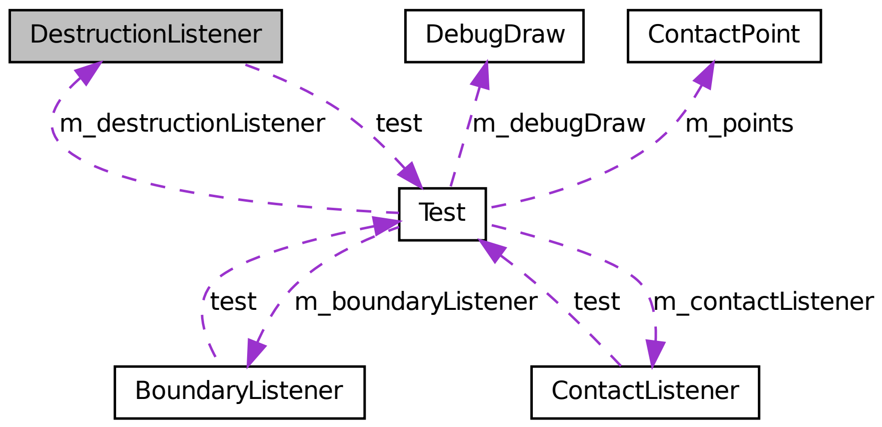
- void **DrawPolygon** (const b2Vec2 ∗vertices, int32 vertexCount, const b2Color &color)
- void **DrawSolidPolygon** (const b2Vec2 ∗vertices, int32 vertexCount, const b2Color &color)
- void **DrawCircle** (const b2Vec2 &center, float32 radius, const b2Color &color)
- void **DrawSolidCircle** (const b2Vec2 &center, float32 radius, const b2Vec2 &axis, const b2Color &color)
- void **DrawSegment** (const b2Vec2 &p1, const b2Vec2 &p2, const b2Color &color)
- void **DrawXForm** (const b2XForm &xf)

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

- Framework/Render.h
- Framework/Render.cpp

4.6 DestructionListener Class Reference

Collaboration diagram for DestructionListener:



Public Member Functions

- void **SayGoodbye** (b2Shape ∗shape)
- void **SayGoodbye** (b2Joint ∗joint)

Public Attributes

- Test ∗ **test**

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

- Framework/Test.h
- Framework/Test.cpp

4.7 Math Class Reference

```
#include <math.h>
```

Public Member Functions

- float uniformRandom (void)
- float uniformRandom (float lower, float upper)
- void seed (void)

  Seeds the random generator.

- void swap (float &f1, float &f2)
- float approxEuclidean (float, float)

  Calculate the approximation of the Euclidian distance.

4.7.1 Detailed Description

Math class

4.7.2 Member Function Documentation

4.7.2.1 void Math::swap (float & *f1*, float & *f2*)

Swaps two values with each other

Parameters

`out`	*f1*	value one
`out`	*f2*	value two

4.7.2.2 float Math::uniformRandom (void)

Returns

A uniform random value between 0 and 1

4.7.2.3 float Math::uniformRandom (float *lower*, float *upper*)

Parameters

`in`	*lower*	the lower bound
`in`	*upper*	the upper bound

Returns

A uniform random value between lower and upper

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

- ai/math.h

- ai/math.cpp

4.8 Neural Class Reference

Strongly connected recurrent neural network with mutiple inputs and outputs.

`#include <neural.h>`

Public Member Functions

- Neural ()
- Neural (int inputs, int outputs, int hiddenLayers, int nodesPerLayer)
- ∼Neural ()

  Destructor.

- void perform (double ∗inputs, double ∗outputs)
- void setWeights (double ∗∗weightsInputs, double ∗∗weightOutputs, double ∗∗weightsHidden)
- void setWeights (double ∗weights)

  same as above, but this one accepts a serialized array of floats

- int size ()

  Total weights count.

- void print ()

  Prints the whole network to stdout.

4.8.1 Detailed Description

Strongly connected recurrent neural network with mutiple inputs and outputs. This RNN has the form: I -> H1 ... Hn -> O where I is an input layer and O the output. The hidden layer H's are strongly connected. Meaning all the nodes in the the layer has a connection to all other nodes. N nodes per layer are constant. The implementation idea is as follow:

```
Take a network with 2 inputs, 2 outputs, 2 hidden layers and 2 nodes
per layer. It is represented as:
For the nodes:
I = [i1, i2]
O = [o1, o2]
H = [n11, n12, n21, n22]

Their respective weights:
wI = [
   i1 n11, i1 n12, i1 n21, i1 n22;
   i2 n11, i2 n12, i2 n21, i2 n22;
   ]
wO = [
   o1 n11, o1 n12, o1 n21, o1 n22;
   o2 n11, o2 n12, o2 n21, o2 n22;
```

```
    ]
wH = [
   n11 n11, n11 n12, n11 n21, n11 n22;
   n12 n11, n12 n12, ... ;
   ... ;
   n22 n11, ..., n22 n22;
   ]
```

The hidden layers activations are processed one after another. for H1, the first hidden layer:

```
aH(y, x) = sum(Ii*wI(i, n1x)) + sum(aH(y,x) * wH(y, nyx))
aO(y, x) = sum(i*wI(i, n1x) + aH(1,x)
```

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Neural::Neural ( )

Construct a recurrent neural network using default parameters

4.8.2.2 Neural::Neural (int *inputs*, int *outputs*, int *hiddenLayers*, int *nodesPerLayer*)

Construct a RNN using the following parameters

Parameters

in	*inputs*	Number of inputs
in	*outputs*	Number of outputs
in	*hiddenLayers*	Number of hidden layers
in	*nodesPerLayer*	Number of nodes per hidden layer

4.8.3 Member Function Documentation

4.8.3.1 void Neural::perform (double * *inputs*, double * *outputs*)

Calculates the outputs given a set of inputs and the weights for each edge. Number of input elements should match with the amount used to initialize this Neural network. Same for outputs.

Note

This method should be called after setting the weights!

Parameters

in	*inputs*	Set of input values for the network
out	*outputs*	Array where the output values are stored in

4.8.3.2 void Neural::setWeights (double $**$ *weightsInputs*, double $**$ *weightOutputs*, double $**$ *weightsHidden*)

Sets the weight values for all edges. The amount should adhere to (LN)^2+ILN + OLN where L is the number of hidden layers, N nodes per layer, I number of inputs, O number of outputs.

Parameters

`in`	*weightsHidden*	Set of weights for all edges in the hidden layers
`in`	*weightsInputs*	Set of weights for all edges from inputs to all nodes in all hidden layers
`in`	*weightsOutputs*	Set of weights for all edges from all nodes in all hidden layers to all outputs

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

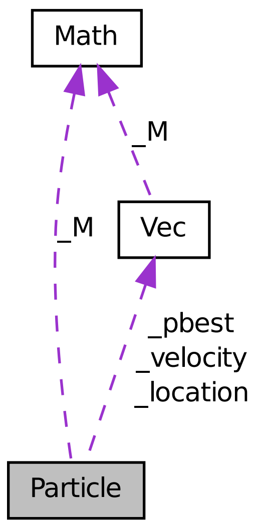
- ai/neural.h
- ai/neural.cpp

4.9 Particle Class Reference

A particle used in Particle Swarm Optimzer.

`#include <particle.h>`

Collaboration diagram for Particle:



Public Member Functions

- Particle (int dimensions, Test ∗test)
- ∼Particle (void)

  Destructs instance.

- void update (Vec gbest, Test ∗test)
- Vec getLocation (void)
- float GetPBestValue ()
- Vec GetPBest ()
- float getFitness ()

4.9.1 Detailed Description

A particle used in Particle Swarm Optimzer. Particle class

Each particle has a position in a predefined dimension d. The current location can be requested, new velocity can be calculated based on PSO

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Particle::Particle (int *dimensions*, Test * *test*)

Constructs a particle

Parameters

`in`	*dimensions*	the dimensionality of this particle
`in`	*test*	the environment

4.9.3 Member Function Documentation

4.9.3.1 float Particle::getFitness ( )

Fitness of this particle. Only call this after calling update()

Returns

Fitness of particle

4.9.3.2 Vec Particle::getLocation (void)

Current particle's location

Returns

particle location

4.9.3.3 Vec Particle::GetPBest ( )

Personal best vector

Returns

personal best vector

4.9.3.4 float Particle::GetPBestValue ( )

Personal best fitness

Returns

personal best fitness

4.9.3.5 void Particle::update (Vec *gbest,* Test ∗ *test*)

Updates particle's velocity and position

Parameters

`in`	*gbest*	The global best so far
`in`	*test*	the environment

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

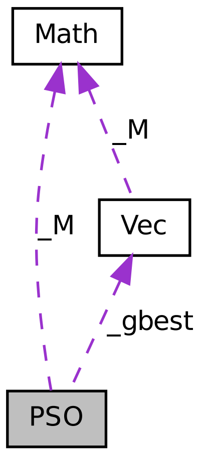
- ai/particle.h
- ai/particle.cpp

4.10 PSO Class Reference

This is a standard Particle Swarm Optimizer that minimizes the fitness.

`#include <pso.h>`

Collaboration diagram for PSO:



Public Member Functions

- PSO (int n, int d, Test ∗test)
- ∼PSO (void)

Destructs the instance.

- void update (Test ∗test)
- Vec GetGBest ()
- float GetGBestValue ()

4.10.1 Detailed Description

This is a standard Particle Swarm Optimizer that minimizes the fitness. PSO class

This class was taken from SBI_DEMO and modified.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 PSO::PSO (int *n*, int *d*, Test ∗ *test*)

Constructs a swarm of specified size

Parameters

in	*n*	number of particles in this population
in	*d*	dimensionality of each particle
in	*test*	the environment

4.10.3 Member Function Documentation

4.10.3.1 Vec PSO::GetGBest ( )

Determine the global best of current population

Returns

global best particle

4.10.3.2 float PSO::GetGBestValue ( )

Fitness of global best, call this after calling update()

Returns

global best fitness

4.10.3.3 void PSO::update (Test ∗ *test*)

Executes an iteration

Parameters

`in`	*test*	the environment

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

- ai/pso.h
- ai/pso.cpp

4.11 Settings Struct Reference

Public Attributes

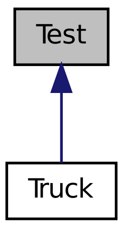
- float32 **hz**
- int32 **iterationCount**
- int32 **drawShapes**
- int32 **drawJoints**
- int32 **drawCoreShapes**
- int32 **drawAABBs**
- int32 **drawOBBs**
- int32 **drawPairs**
- int32 **drawContactPoints**
- int32 **drawContactNormals**
- int32 **drawContactForces**
- int32 **drawFrictionForces**
- int32 **drawCOMs**
- int32 **drawStats**
- int32 **enableWarmStarting**
- int32 **enablePositionCorrection**
- int32 **enableTOI**
- int32 **pause**
- int32 **singleStep**

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

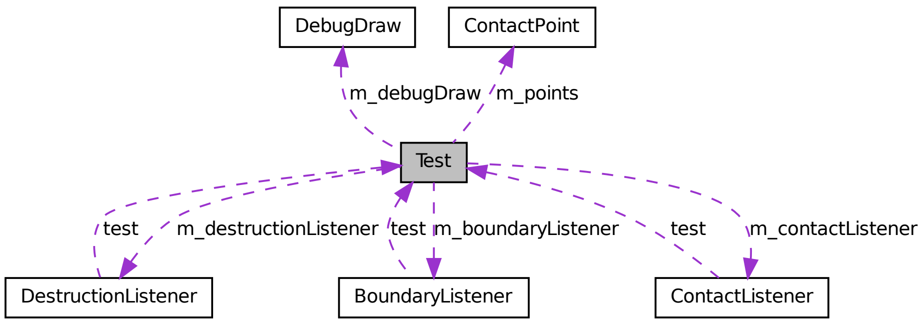
- Framework/Test.h

4.12 Test Class Reference

Inheritance diagram for Test:



Collaboration diagram for Test:



Public Member Functions

- void **SetTextLine** (int32 line)
- virtual void **Step** (Settings ∗settings)
- virtual void **Keyboard** (unsigned char key)
- void **MouseDown** (const b2Vec2 &p)
- void **MouseUp** ()
- void **MouseMove** (const b2Vec2 &p)
- void **LaunchBomb** ()
- virtual void **JointDestroyed** (b2Joint ∗joint)

- virtual void **BoundaryViolated** (b2Body ∗body)

Protected Attributes

- b2AABB **m_worldAABB**
- ContactPoint **m_points** [k_maxContactPoints]
- int32 **m_pointCount**
- DestructionListener **m_destructionListener**
- BoundaryListener **m_boundaryListener**
- ContactListener **m_contactListener**
- DebugDraw **m_debugDraw**
- int32 **m_textLine**
- b2World ∗ **m_world**
- b2Body ∗ **m_bomb**
- b2MouseJoint ∗ **m_mouseJoint**

Friends

- class DestructionListener
- class BoundaryListener
- class ContactListener

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

- Framework/Test.h
- Framework/Test.cpp

4.13 TestEntry Struct Reference

Public Attributes

- const char ∗ **name**
- TestCreateFcn ∗ **createFcn**

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

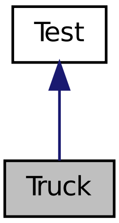
- Framework/Test.h

4.14 Truck Class Reference

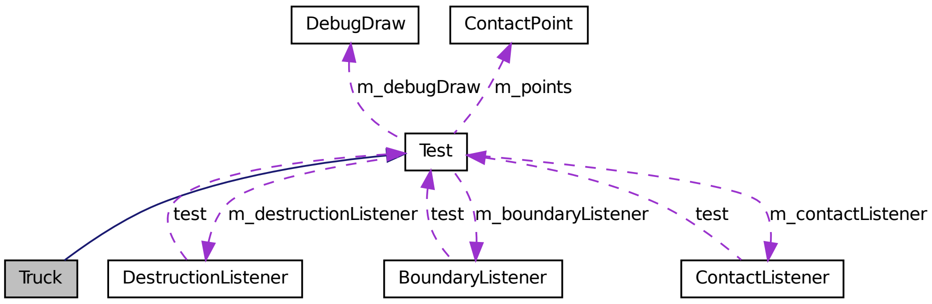
This class defines a special Truck :P.

`#include <Truck.h>`

Inheritance diagram for Truck:



Collaboration diagram for Truck:



Public Member Functions

- Truck ()

  Constructs a Truck.

- void Step (Settings ∗settings)
- void Keyboard (unsigned char key)
- b2Vec2 getBodyOffset ()

- void getEnvironment (double ∗env)
- void simulate (float tstep, int iter)
- void setParameters (double ∗params)
- b2Vec2 getPosition ()

Static Public Member Functions

- static Test ∗ Create ()

  Creates an instance of Truck.

4.14.1 Detailed Description

This class defines a special Truck :P. This very special truck was ported from flash[1] to C++. "Sensors" and actuators are configurable using the global config[] variable.

1. `http://www.emanueleferonato.com/2009/04/06/two-ways-to-make-box2d-cars/`

4.14.2 Member Function Documentation

4.14.2.1 b2Vec2 Truck::getBodyOffset ( )

The initial position of the vehicle

Returns

initial position

4.14.2.2 void Truck::getEnvironment (double ∗ *env*)

Reads sensor data. The number of variables depend on config[CONFIG_IN_NODES]

Parameters

`out`	*env*	values are read into env

4.14.2.3 b2Vec2 Truck::getPosition ( )

Returns

current position of the vehicle

4.14.2.4 void Truck::Keyboard (unsigned char *key*) `[virtual]`

React to some keystrokes.

Note

There's delay in registering the strokes when being used in visualization mode

Parameters

key]	the keystroke value

Reimplemented from Test.

4.14.2.5 void Truck::setParameters (double ∗ *params*)

Sets the actuator values

Parameters

in	*params*	set of actuator values

4.14.2.6 void Truck::simulate (float *tstep,* int *iter*)

Simulates a step, not really used by the algorithm

Parameters

in	*tstep*	time step
in	*iter*	iterations

4.14.2.7 void Truck::Step (Settings ∗ *settings*) `[virtual]`

Executes a step in the world

Parameters

in	*settings*	using settings

Reimplemented from Test.

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

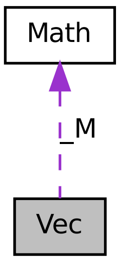
- ai/Truck.h
- ai/Truck.cpp

4.15 Vec Class Reference

Implements a special type of vector.

`#include <vec.h>`

Collaboration diagram for Vec:



Public Member Functions

- **Vec** (vector< float >)
- Vec **operator=** (vector< float >)
- Vec **operator=** (Vec)
- Vec **operator*** (float)
- Vec **operator/** (float)
- Vec **operator+** (vector< float >)
- Vec **operator+** (Vec)
- Vec **operator-** (vector< float >)
- Vec **operator-** (Vec)
- Vec **normalized** (void)
- float **magnitude** (void)
- void **print** (void)
- void **print** (string)
- void **printMagnitude** (void)
- void **printMagnitude** (string)
- Vec **enforceVMax** (void)
- Vec **enforceXMax** (void)
- vector< float > **GetWeights** ()

4.15.1 Detailed Description

Implements a special type of vector. Vec class

one with operators that can be applied to each individual element

This class was taken from SBI_DEMO and modified

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

- ai/vec.h
- ai/vec.cpp

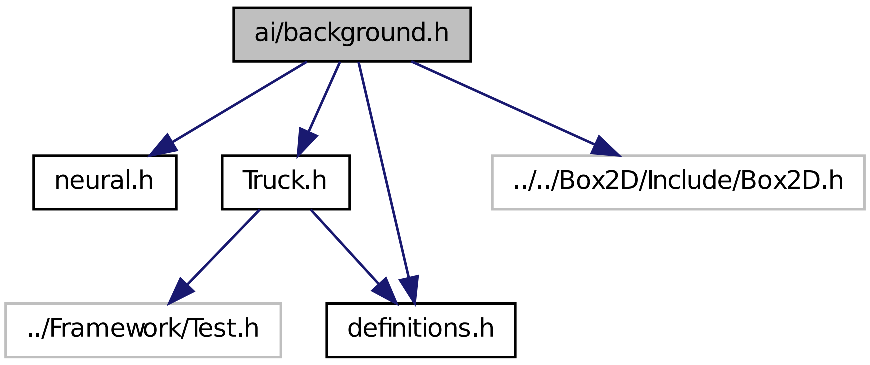
Chapter 5

File Documentation

5.1 ai/background.h File Reference

```
#include "neural.h"
#include "Truck.h"
#include "../../Box2D/Include/Box2D.h"
#include "definitions.h"
```

Include dependency graph for background.h:



Classes

- class Background

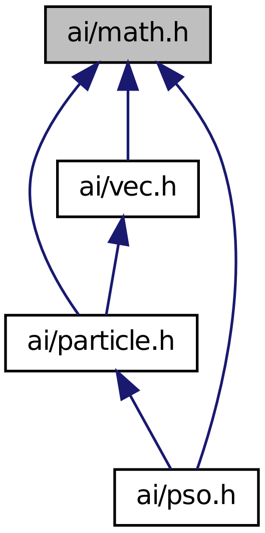
5.1.1 Detailed Description

Author

xcheng, bvveen, unknown

5.2 ai/math.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Math

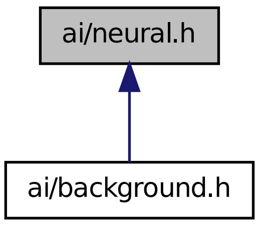
5.2.1 Detailed Description

Author

unknown

5.3 ai/neural.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Neural

  Strongly connected recurrent neural network with mutiple inputs and outputs.

5.3.1 Detailed Description

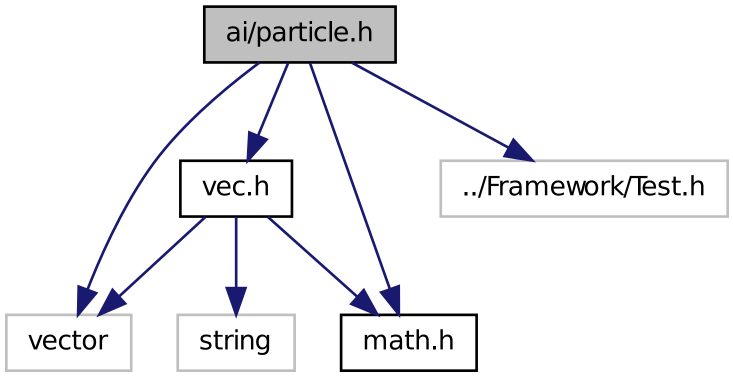
Author

xcheng

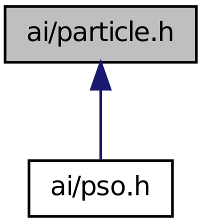
5.4 ai/particle.h File Reference

```
#include <vector>
#include "math.h"
#include "vec.h"
#include "../Framework/Test.h"
```

Include dependency graph for particle.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Particle

  A particle used in Particle Swarm Optimzer.

5.4.1 Detailed Description

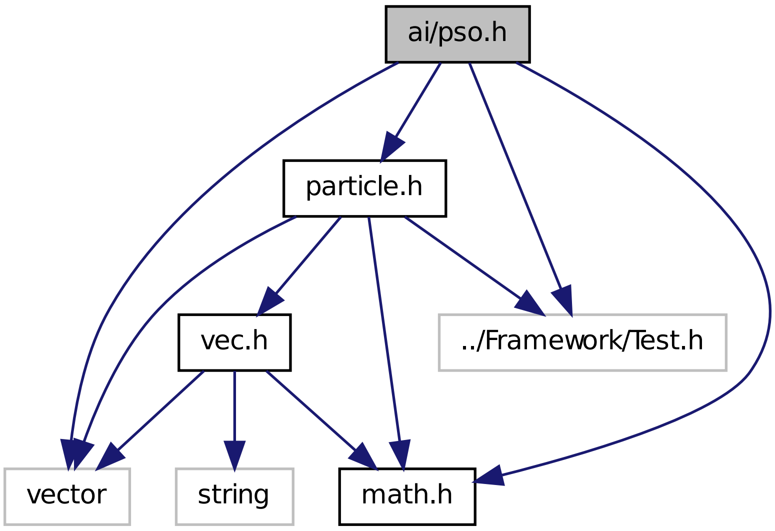
Author

bvveen, xcheng, unknown

5.5 ai/pso.h File Reference

```
#include "particle.h"
#include "math.h"
#include "../Framework/Test.h"
#include <vector>
```

Include dependency graph for pso.h:



Classes

- class PSO

  This is a standard Particle Swarm Optimizer that minimizes the fitness.

5.5.1 Detailed Description

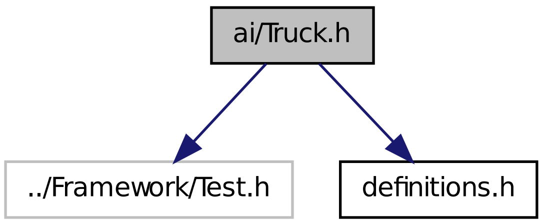
Author

bvveen, xcheng, unknown

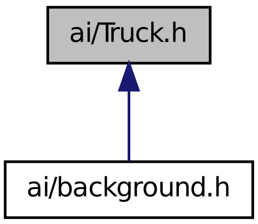
5.6 ai/Truck.h File Reference

```
#include "../Framework/Test.h"
#include "definitions.h"
```

Include dependency graph for Truck.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Truck

This class defines a special Truck :P.

5.6.1 Detailed Description

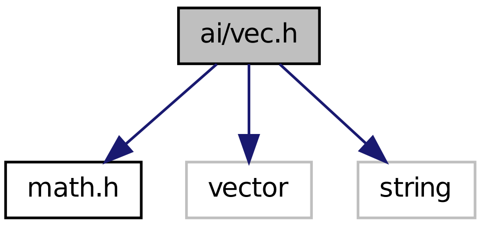
Author

xcheng, bvveen

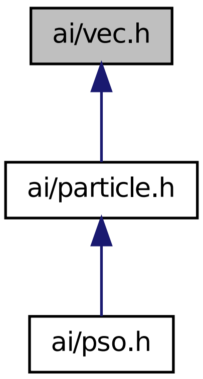
5.7 ai/vec.h File Reference

```
#include "math.h"
#include <vector>
#include <string>
```

Include dependency graph for vec.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Vec

  Implements a special type of vector.

5.7.1 Detailed Description

Author

xcheng, unknown

Index