Starlings

1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

starlings

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 boid Class Reference

Public Member Functions

- velocity_struct cohesion (vector< boid > &arr)
- velocity_struct alignment (vector< boid > &arr)
- velocity_struct separation (vector< boid > &arr)
- velocity_struct main_rule (vector< boid > &arr, vector< obstacle > &obs)
- velocity_struct bound_position ()
- velocity_struct goal_seeking (position_struct &goal_pos)
- void limit_velocity ()
- boid (float radius=ra, float m=5.0, float min_dis=md)
- boid (position_struct p, velocity_struct v, float radius=ra, float m=5.0, float min_dis=md)
- position_struct get_position ()
- velocity_struct get_velocity ()
- float get_mass ()
- void set_position (position_struct &p)
- void set_velocity (velocity_struct &v)
- void add_velocity (velocity_struct &v)
- void add_position (velocity_struct &p)
- velocity_struct get_update (vector< boid > &flk, std::vector< obstacle > &obsts, position_struct &goal_pos, velocity_struct &wind)
- void print ()

Public Attributes

- position_struct pos
- velocity_struct vel
- float radius_of_influence
- float mass
- float minimum_distance
- position_struct max_position
- float max_speed

3.1.1 Constructor & Destructor Documentation

3.1.1.1 boid::boid (float *radius* = `ra`, float *m* = `5.0`, float *min_dis* = `md`)

Create a new boid

Parameters

radius	radius of influnce
m	mass
min_dis	minimum distance

3.1.1.2 boid::boid (position_struct *p*, velocity_struct *v*, float *radius* = ra, float *m* = 5.0, float *min_dis* = md)

Create a new boid

Parameters

p	initial position
v	initial velocity
radius	radius of influnce
m	mass
min_dis	minimum distance

3.1.2 Member Function Documentation

3.1.2.1 void boid::add_position (velocity_struct & *p*)

adds v to current position

Parameters

v	position to be added to current position

3.1.2.2 void boid::add_velocity (velocity_struct & *v*)

adds v to current velocity

Parameters

v	velocity to be added to current velocity

3.1.2.3 velocity_struct boid::alignment (vector< boid > & *arr*)

returns velocity direction due to alignment property

3.1.2.4 velocity_struct boid::bound_position ( )

returns velocity direction for bounding position

3.1.2.5 velocity_struct boid::cohesion (vector< boid > & *arr*)

returns velocity direction due to cohesion

3.1.2.6 float boid::get_mass ( )

returns mass of the boid

3.1.2.7 position_struct boid::get_position ( )

returns current position of the boid

3.1.2.8 velocity_struct boid::get_update (vector< boid > & *flk,* std::vector< obstacle > & *obsts,* position_struct & *goal_pos,* velocity_struct & *wind*)

calculates the force to be applied to the boid

Parameters

flk	vector containing all boids
obsts	vector containing all obstacles
goal_pos	position of goal
wind	velocity due to wind

3.1.2.9 velocity_struct boid::get_velocity ( )

returns current velocity of the boid

3.1.2.10 velocity_struct boid::goal_seeking (position_struct & *goal_pos*)

returns velocity for goal seeking behaviour

3.1.2.11 void boid::limit_velocity ( )

limits velocity to max_speed

3.1.2.12 velocity_struct boid::main_rule (vector< boid > & *arr,* vector< obstacle > & *obs*)

combines all rules in a single function

3.1.2.13 void boid::print ( )

prints current position and velocity of the boid

3.1.2.14 velocity_struct boid::separation (vector< boid > & *arr*)

returns velocity direction due to seperation property

3.1.2.15 void boid::set_position (position_struct & *p*)

sets current position of boid to p

Parameters

p	new position

3.1.2.16 void boid::set_velocity (velocity_struct & *v*)

sets current velocity of boid to p

Parameters

p	new velocity

3.1.3 Member Data Documentation

3.1.3.1 float boid::mass

mass of boid

3.1.3.2 position_struct boid::max_position

Boundary of the world (assumed to be a cuboid)

3.1.3.3 float boid::max_speed

Maximum speed a boid can fly with

3.1.3.4 float boid::minimum_distance

minimum distance between two boids

3.1.3.5 position_struct boid::pos

current position of boid

3.1.3.6 float boid::radius_of_influence

distance upto which boid can see

3.1.3.7 velocity_struct boid::vel

current velocity of boid

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

- include/boid.h
- src/boid.cpp

3.2 flock Class Reference

Public Member Functions

- flock ()
- void **initialize** ()
- int getTotalBoids ()
- vector< boid > getBoids ()
- void add_boid ()
- void add_boid (boid b)
- void spawn (int n=1000)
- int getTotalObstacles ()
- void set_goal (position_struct &p)
- void update_goal ()
- void add_obstacle ()
- void add_obstacle (obstacle o)
- void update ()
- void render ()
- void **new_render** ()

Public Attributes

- vector< boid > **flk**
- vector< obstacle > **obsts**
- position_struct **goal_pos**
- velocity_struct **wind**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 flock::flock ( )

creates a flock

3.2.2 Member Function Documentation

3.2.2.1 void flock::add_boid ( )

adds a boid to the flock

3.2.2.2 void flock::add_boid (boid *b*)

adds a boid b to the flock

Parameters

b	boid to be added to the flock

3.2.2.3 void flock::add_obstacle ( )

adds an obstacle to the flock

3.2.2.4 void flock::add_obstacle (obstacle *o*)

add an obstacle o to the flock

3.2.2.5 vector< boid > flock::getBoids ( )

returns a vector containing all boids in the flock

3.2.2.6 int flock::getTotalBoids ( )

returns total number of boids in tbe flock

3.2.2.7 int flock::getTotalObstacles ( )

returns total number of obstacles

3.2.2.8 void flock::render ( )

renders the flock using OpenGl

3.2.2.9 void flock::set_goal (position_struct & *p*)

sets the position of goal to p

3.2.2.10 void flock::spawn (int *n* = 1000)

creates n boids with random initial positions and velocities

Parameters

n	number of boids

3.2.2.11 void flock::update ( )

updates position and velocities of all the boids in the flock

3.2.2.12 void flock::update_goal ( )

updates goal position randomly

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

- include/flock.h
- src/flock.cpp

3.3 obstacle Class Reference

Public Member Functions

- obstacle ()
- obstacle (position_struct p, float m=0.0)\
- position_struct get_position ()
- float get_mass ()

Public Attributes

- position_struct **pos**
- float **mass**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 obstacle::obstacle ( )

Creates an empty obstacle

3.3.1.2 obstacle::obstacle (position_struct *p*, float *m* = `0.0`)

Creates an obstacle at given position and of given mass

Parameters

p	position of obstacle
m	mass of obstacle

3.3.2 Member Function Documentation

3.3.2.1 float obstacle::get_mass ( )

Returns mass of the obstacle

3.3.2.2 position_struct obstacle::get_position ( )

Returns the position of the obstacle

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

- include/boid.h
- src/boid.cpp

3.4 position_struct Struct Reference

`#include <ds.h>`

Public Member Functions

- position_struct ()
- position_struct (float a, float b, float c)
- void add (position_struct &p)
- void add (position_struct &v)
- void add (float a, float b, float c)
- void subtract (float a, float b, float c)
- void divide (float a, float b, float c)
- void divide (float a)
- void multiply (float a)
- void set (position_struct p)
- void normalise ()

Public Attributes

- float **x**
- float **y**
- float **z**

3.4.1 Detailed Description

To store position of the boid

3.4.2 Constructor & Destructor Documentation

3.4.2.1 position_struct::position_struct ( ) [inline]

initializes position to zero

3.4.2.2 position_struct::position_struct (float *a,* float *b,* float *c*) [inline]

initializes position to given value

Parameters

a	x component of position
b	y component of position
c	z component of position

3.4.3 Member Function Documentation

3.4.3.1 void position_struct::add (position_struct & *p*) [inline]

adds position p to current position

Parameters

p	position to be added to current position

3.4.3.2 void position_struct::add (position_struct & *v*) [inline]

adds velocity v to current position

Parameters

v	velocity to be added to current position

3.4.3.3 void position_struct::add (float *a,* float *b,* float *c*) [inline]

adds position gicen to current position

Parameters

a	x component of position
b	y component of position
c	z component of position

3.4.3.4 void position_struct::divide (float *a,* float *b,* float *c*) [inline]

divides position by different values in different directions

Parameters

a	x component of position
b	y component of position
c	z component of position

3.4.3.5 void position_struct::divide (float *a*) [inline]

divides position by a uniform value in all different directions

Parameters

a	dividing factor

3.4.3.6 void position_struct::multiply (float *a*) [inline]

multiplies position by different values in different directions

Parameters

a	multiplying factors

3.4.3.7 void position_struct::normalise ( ) [inline]

converts speed to unity retaining same direction

3.4.3.8 void position_struct::set (position_struct *p*) [inline]

sets position to given value

Parameters

new	position

3.4.3.9 void position_struct::subtract (float *a*, float *b*, float *c*) [inline]

subtract position gicen to current position

Parameters

a	x component of position
b	y component of position
c	z component of position

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

- include/ds.h

3.5 velocity_struct Struct Reference

`#include <ds.h>`

Public Member Functions

- velocity_struct ()
- velocity_struct (float a, float b, float c)
- void add (velocity_struct &v)
- void add (float a, float b, float c)
- void subtract (velocity_struct &v)
- void subtract (float a, float b, float c)
- void divide (float a, float b, float c)
- void divide (float a)
- void multiply (float a)
- void set (velocity_struct &v)
- void **set** (float a, float b, float c)
- void normalise ()

Public Attributes

- float **x**
- float **y**
- float **z**

3.5.1 Detailed Description

To store velocity of the boid

3.5.2 Constructor & Destructor Documentation

3.5.2.1 velocity_struct::velocity_struct ( ) [inline]

initializes velocity to zero

3.5.2.2 velocity_struct::velocity_struct (float *a,* float *b,* float *c*) [inline]

initializes velocity to given value

Parameters

a	x component of velocity
b	y component of velocity
c	z component of velocity

3.5.3 Member Function Documentation

3.5.3.1 void velocity_struct::add (velocity_struct & *v*) [inline]

adds velocity v to current velocity

Parameters

v	velocity to be added to current velocity

3.5.3.2 void velocity_struct::add (float *a,* float *b,* float *c*) [inline]

adds velocity gicen to current velocity

Parameters

a	x component of velocity
b	y component of velocity
c	z component of velocity

3.5.3.3 void velocity_struct::divide (float *a,* float *b,* float *c*) [inline]

divides velocity by different values in different directions

Parameters

a	x component of velocity
b	y component of velocity
c	z component of velocity

3.5.3.4 void velocity_struct::divide (float *a*) [inline]

divides velocity by a uniform value in all different directions

Parameters

a	dividing factor

3.5.3.5 void velocity_struct::multiply (float *a*) [inline]

multiplies velocity by different values in different directions

Parameters

a	multiplying factors

3.5.3.6 void velocity_struct::normalise ( ) [inline]

converts distance to unity retaining same direction

3.5.3.7 void velocity_struct::set (velocity_struct & *v*) [inline]

sets position to given value

Parameters

v	new position

3.5.3.8 void velocity_struct::subtract (velocity_struct & *v*) [inline]

subtracts velocity v to current velocity

Parameters

v	velocity to be subtracted to current velocity

3.5.3.9 void velocity_struct::subtract (float *a*, float *b*, float *c*) [inline]

subtract velocity gicen to current velocity

Parameters

a	x component of velocity

Parameters

b	y component of velocity
c	z component of velocity

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

- include/ds.h

Index