

Autonomous Steering Agents

Generated by Doxygen 1.8.17

1 Intent	1
1.1 Dependencies	1
1.2 Resources	1
2 Hierarchical 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 agent Class Reference	9
5.1.1 Detailed Description	9
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 agent() [1/2]	10
5.1.2.2 agent() [2/2]	10
5.1.2.3 ~agent()	10
5.1.3 Member Function Documentation	10
5.1.3.1 setFeatures()	10
5.1.3.2 updatePosition()	11
5.1.4 Member Data Documentation	11
5.1.4.1 acceleration	11
5.1.4.2 arrive	11
5.1.4.3 desiredVelocity	11
5.1.4.4 fillColor	12
5.1.4.5 force	12
5.1.4.6 id	12
5.1.4.7 mass	12
5.1.4.8 maxForce	12
5.1.4.9 maxSpeed	12
5.1.4.10 name	13
5.1.4.11 position	13
5.1.4.12 r	13
5.1.4.13 steering	13
5.1.4.14 targetPoint	13
5.1.4.15 velocity	13
5.2 color Class Reference	14
5.2.1 Detailed Description	14
5.2.2 Constructor & Destructor Documentation	14
5.2.2.1 color() [1/2]	14
5.2.2.2 color() [2/2]	15

5.2.3 Member Function Documentation	15
5.2.3.1 createColors()	15
5.2.3.2 getColor()	15
5.2.4 Member Data Documentation	16
5.2.4.1 B	16
5.2.4.2 colors	16
5.2.4.3 G	16
5.2.4.4 R	17
5.3 evade Class Reference	17
5.3.1 Detailed Description	17
5.3.2 Constructor & Destructor Documentation	17
5.3.2.1 evade()	18
5.3.3 Member Function Documentation	18
5.3.3.1 loop()	18
5.4 flee Class Reference	18
5.4.1 Detailed Description	19
5.4.2 Constructor & Destructor Documentation	19
5.4.2.1 flee()	19
5.4.3 Member Function Documentation	19
5.4.3.1 loop()	19
5.5 flock Class Reference	19
5.5.1 Detailed Description	20
5.5.2 Constructor & Destructor Documentation	20
5.5.2.1 flock()	20
5.5.3 Member Function Documentation	20
5.5.3.1 loop()	20
5.6 flowField Class Reference	21
5.6.1 Detailed Description	21
5.6.2 Constructor & Destructor Documentation	21
5.6.2.1 flowField() [1/2]	21
5.6.2.2 flowField() [2/2]	21
5.6.3 Member Function Documentation	22
5.6.3.1 getField()	22
5.7 graphics Class Reference	22
5.7.1 Detailed Description	23
5.7.2 Member Function Documentation	23
5.7.2.1 drawAgent()	23
5.7.2.2 drawCircle()	24
5.7.2.3 drawLine()	24
5.7.2.4 drawPath()	24
5.7.2.5 drawPoint()	25
5.7.2.6 drawText()	25

5.7.2.7 drawWall()	25
5.7.2.8 forceInScreen()	26
5.7.2.9 getMousePosition()	26
5.7.2.10 handleKeypress()	26
5.7.2.11 handleResize()	26
5.7.2.12 initGraphics()	27
5.7.2.13 mouseButton()	27
5.7.2.14 mouseMove()	27
5.7.2.15 refreshScene()	28
5.7.2.16 timerEvent()	28
5.7.3 Member Data Documentation	28
5.7.3.1 target_x	28
5.7.3.2 target_y	28
5.8 mouseFollower Class Reference	29
5.8.1 Detailed Description	29
5.8.2 Constructor & Destructor Documentation	29
5.8.2.1 mouseFollower()	29
5.8.3 Member Function Documentation	29
5.8.3.1 loop()	30
5.9 obstacle Class Reference	30
5.9.1 Detailed Description	30
5.9.2 Constructor & Destructor Documentation	30
5.9.2.1 obstacle() [1/2]	31
5.9.2.2 obstacle() [2/2]	31
5.9.3 Member Data Documentation	31
5.9.3.1 p	31
5.9.3.2 r	32
5.10 obstacleAvoidance Class Reference	32
5.10.1 Detailed Description	32
5.10.2 Constructor & Destructor Documentation	32
5.10.2.1 obstacleAvoidance()	33
5.10.3 Member Function Documentation	33
5.10.3.1 createObstacle()	33
5.10.3.2 loop()	33
5.10.4 Member Data Documentation	33
5.10.4.1 obstacles	34
5.11 path Class Reference	34
5.11.1 Detailed Description	34
5.11.2 Constructor & Destructor Documentation	34
5.11.2.1 path() [1/2]	35
5.11.2.2 path() [2/2]	35
5.11.3 Member Function Documentation	35

5.11.3.1 addPoint()	35
5.11.4 Member Data Documentation	36
5.11.4.1 points	36
5.11.4.2 width	36
5.12 pathFollower Class Reference	36
5.12.1 Detailed Description	37
5.12.2 Constructor & Destructor Documentation	37
5.12.2.1 pathFollower()	37
5.12.3 Member Function Documentation	37
5.12.3.1 createPath()	38
5.12.3.2 loop()	38
5.12.4 Member Data Documentation	38
5.12.4.1 myPath	38
5.13 point Class Reference	38
5.13.1 Detailed Description	39
5.13.2 Constructor & Destructor Documentation	39
5.13.2.1 point() [1/2]	39
5.13.2.2 point() [2/2]	39
5.13.3 Member Function Documentation	40
5.13.3.1 div()	40
5.13.3.2 getNormalPoint()	40
5.13.3.3 mul()	40
5.13.3.4 operator+() [1/2]	40
5.13.3.5 operator+() [2/2]	41
5.13.3.6 operator-()	41
5.13.3.7 operator==(())	41
5.13.3.8 print()	41
5.13.4 Member Data Documentation	41
5.13.4.1 x	42
5.13.4.2 y	42
5.14 prison Class Reference	42
5.14.1 Detailed Description	42
5.14.2 Constructor & Destructor Documentation	42
5.14.2.1 prison()	43
5.14.3 Member Function Documentation	43
5.14.3.1 loop()	43
5.15 pursuit Class Reference	43
5.15.1 Detailed Description	44
5.15.2 Constructor & Destructor Documentation	44
5.15.2.1 pursuit()	44
5.15.3 Member Function Documentation	44
5.15.3.1 loop()	44

5.16 pvector Class Reference	44
5.16.1 Detailed Description	45
5.16.2 Constructor & Destructor Documentation	45
5.16.2.1 pvector() [1/2]	45
5.16.2.2 pvector() [2/2]	45
5.16.3 Member Function Documentation	46
5.16.3.1 add()	46
5.16.3.2 angleBetween()	46
5.16.3.3 div()	46
5.16.3.4 dotProduct()	46
5.16.3.5 getAngle()	47
5.16.3.6 limit()	47
5.16.3.7 magnitude()	47
5.16.3.8 mul()	47
5.16.3.9 normalize()	48
5.16.3.10 operator+() [1/2]	48
5.16.3.11 operator+() [2/2]	48
5.16.3.12 operator+=()	48
5.16.3.13 operator-() [1/2]	49
5.16.3.14 operator-() [2/2]	49
5.16.3.15 operator==()	49
5.16.3.16 print()	49
5.16.4 Member Data Documentation	49
5.16.4.1 x	50
5.16.4.2 y	50
5.17 random Class Reference	50
5.17.1 Detailed Description	50
5.17.2 Member Function Documentation	50
5.17.2.1 createRandomArray()	51
5.18 scenario Class Reference	51
5.18.1 Detailed Description	52
5.18.2 Constructor & Destructor Documentation	52
5.18.2.1 scenario()	52
5.18.3 Member Function Documentation	52
5.18.3.1 createAgent()	52
5.18.3.2 initGL()	52
5.18.3.3 refresh()	53
5.18.4 Member Data Documentation	53
5.18.4.1 agents	53
5.18.4.2 behavior	53
5.18.4.3 callback	53
5.18.4.4 myColor	53

5.18.4.5 name	54
5.18.4.6 view	54
5.19 steeringBehavior Class Reference	54
5.19.1 Detailed Description	54
5.19.2 Member Function Documentation	55
5.19.2.1 align()	55
5.19.2.2 avoid()	55
5.19.2.3 cohesion()	56
5.19.2.4 evade()	56
5.19.2.5 flee()	57
5.19.2.6 inFlowField()	57
5.19.2.7 pursuit()	57
5.19.2.8 seek()	58
5.19.2.9 separation()	58
5.19.2.10 setAngle()	58
5.19.2.11 stayInArea()	59
5.19.2.12 stayInPath()	59
5.19.2.13 stayInPath_2()	59
5.19.2.14 wander()	60
5.20 wander Class Reference	60
5.20.1 Detailed Description	61
5.20.2 Constructor & Destructor Documentation	61
5.20.2.1 wander()	61
5.20.3 Member Function Documentation	61
5.20.3.1 loop()	61
5.21 windy Class Reference	62
5.21.1 Detailed Description	62
5.21.2 Constructor & Destructor Documentation	62
5.21.2.1 windy()	62
5.21.3 Member Function Documentation	62
5.21.3.1 loop()	63
5.21.4 Member Data Documentation	63
5.21.4.1 flow	63
6 File Documentation	65
6.1 include/agent.h File Reference	65
6.2 include/color.h File Reference	65
6.2.1 Detailed Description	65
6.2.2 Enumeration Type Documentation	66
6.2.2.1 num	66
6.3 include/evade.h File Reference	66
6.4 include/flee.h File Reference	66

6.5 include/flock.h File Reference	67
6.6 include/flowField.h File Reference	67
6.6.1 Detailed Description	67
6.6.2 Macro Definition Documentation	67
6.6.2.1 GRAVITY	68
6.6.2.2 HEIGHT	68
6.6.2.3 WIDTH	68
6.6.2.4 WIND_WEST	68
6.7 include/graphics.h File Reference	68
6.7.1 Macro Definition Documentation	69
6.7.1.1 ESC	69
6.7.1.2 HEIGHT	69
6.7.1.3 PI	69
6.7.1.4 WIDTH	69
6.8 include/mouseFollower.h File Reference	69
6.9 include/obstacle.h File Reference	70
6.9.1 Detailed Description	70
6.10 include/obstacleAvoidance.h File Reference	70
6.11 include/path.h File Reference	70
6.11.1 Detailed Description	71
6.12 include/pathFollower.h File Reference	71
6.13 include/point.h File Reference	71
6.14 include/prison.h File Reference	71
6.15 include/pursuit.h File Reference	72
6.16 include/pvector.h File Reference	72
6.16.1 Macro Definition Documentation	72
6.16.1.1 PI	72
6.17 include/random.h File Reference	72
6.18 include/scenario.h File Reference	73
6.18.1 Enumeration Type Documentation	73
6.18.1.1 types	73
6.19 include/steeringBehavior.h File Reference	73
6.19.1 Macro Definition Documentation	74
6.19.1.1 AVOID_OBSTACLE	74
6.19.1.2 CIRCLE_DISTANCE	74
6.19.1.3 CIRCLE_RADIUS	74
6.19.1.4 EVADE	74
6.19.1.5 FLEE	75
6.19.1.6 FLOCK	75
6.19.1.7 FOLLOW_MOUSE	75
6.19.1.8 IN_FLOW_FIELD	75
6.19.1.9 PURSUIT	75

6.19.1.10 STAY_IN_FIELD	75
6.19.1.11 STAY_IN_PATH	76
6.19.1.12 WANDER	76
6.20 include/wander.h File Reference	76
6.21 include/windy.h File Reference	76
6.22 main.cpp File Reference	76
6.22.1 Function Documentation	77
6.22.1.1 main()	77
6.22.1.2 menu()	78
6.22.2 Variable Documentation	78
6.22.2.1 mode	78
6.23 README.md File Reference	78
6.24 src/agent.cpp File Reference	78
6.25 src/color.cpp File Reference	78
6.25.1 Detailed Description	79
6.26 src/evade.cpp File Reference	79
6.27 src/flee.cpp File Reference	79
6.28 src/flock.cpp File Reference	79
6.29 src/flowField.cpp File Reference	79
6.29.1 Detailed Description	80
6.30 src/graphics.cpp File Reference	80
6.31 src/mouseFollower.cpp File Reference	80
6.32 src/obstacle.cpp File Reference	80
6.32.1 Detailed Description	80
6.33 src/obstacleAvoidance.cpp File Reference	81
6.34 src/path.cpp File Reference	81
6.34.1 Detailed Description	81
6.35 src/pathFollower.cpp File Reference	81
6.36 src/point.cpp File Reference	81
6.37 src/prison.cpp File Reference	82
6.38 src/pursuit.cpp File Reference	82
6.39 src/pvector.cpp File Reference	82
6.40 src/random.cpp File Reference	82
6.41 src/scenario.cpp File Reference	82
6.42 src/steeringBehavior.cpp File Reference	83
6.43 src/wander.cpp File Reference	83
6.44 src/windy.cpp File Reference	83
6.45 test/test_suites.cpp File Reference	83
6.45.1 Macro Definition Documentation	84
6.45.1.1 BOOST_TEST_MODULE	84
6.45.2 Function Documentation	84
6.45.2.1 BOOST_AUTO_TEST_CASE() [1/12]	84

6.45.2.2 BOOST_AUTO_TEST_CASE() [2/12]	84
6.45.2.3 BOOST_AUTO_TEST_CASE() [3/12]	85
6.45.2.4 BOOST_AUTO_TEST_CASE() [4/12]	85
6.45.2.5 BOOST_AUTO_TEST_CASE() [5/12]	85
6.45.2.6 BOOST_AUTO_TEST_CASE() [6/12]	85
6.45.2.7 BOOST_AUTO_TEST_CASE() [7/12]	86
6.45.2.8 BOOST_AUTO_TEST_CASE() [8/12]	86
6.45.2.9 BOOST_AUTO_TEST_CASE() [9/12]	86
6.45.2.10 BOOST_AUTO_TEST_CASE() [10/12]	86
6.45.2.11 BOOST_AUTO_TEST_CASE() [11/12]	87
6.45.2.12 BOOST_AUTO_TEST_CASE() [12/12]	87

Index	89
--------------	-----------

Chapter 1

Intent

- 1- implementing Craig Reynolds autonomous steering agents
- 2- implementing genetics algorithms
- 3- implementing neural network

1.1 Dependencies

```
$sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

```
$sudo apt-get install libboost-all-dev
```

1.2 Resources

<https://natureofcode.com/book/chapter-6-autonomous-agents>

<https://gamedevelopment.tutsplus.com/series/understanding-steering-behaviors-gamedev-12>

<https://videotutorialsrock.com/index.php>

<https://www.opengl.org/resources/libraries/glut/spec3/node1.html>

<https://learnopengl.com/Getting-started/Coordinate-Systems>

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

agent	9
color	14
flowField	21
graphics	22
obstacle	30
path	34
point	38
pvector	44
random	50
scenario	51
evade	17
flee	18
flock	19
mouseFollower	29
obstacleAvoidance	32
pathFollower	36
prison	42
pursuit	43
wander	60
windy	62
steeringBehavior	54

Chapter 3

Class Index

3.1 Class List

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

agent	9
color	14
evade	17
flee	18
flock	19
flowField	21
graphics	22
mouseFollower	29
obstacle	30
obstacleAvoidance	32
path	34
pathFollower	36
point	38
prison	42
pursuit	43
pvector	44
random	50
scenario	51
steeringBehavior	54
wander	60
windy	62

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

main.cpp	76
include/agent.h	65
include/color.h	
Color class used for agent, path, wall etc. color	65
include/evade.h	66
include/flee.h	66
include/flock.h	67
include/flowField.h	
FlowField class, screen can be filled with a force for each pixel	67
include/graphics.h	68
include/mouseFollower.h	69
include/obstacle.h	
Circular obstacles for agent avoidance behaviors	70
include/obstacleAvoidance.h	70
include/path.h	
Path class used for path following steering behaviors	70
include/pathFollower.h	71
include/point.h	71
include/prison.h	71
include/pursuit.h	72
include/pvector.h	72
include/random.h	72
include/scenario.h	73
include/steeringBehavior.h	73
include/wander.h	76
include/windy.h	76
src/agent.cpp	78
src/color.cpp	
Color class implementation	78
src/evade.cpp	79
src/flee.cpp	79
src/flock.cpp	79
src/flowField.cpp	
FlowField class implementation	79
src/graphics.cpp	80

src/ mouseFollower.cpp	80
src/ obstacle.cpp	
Obstacle class implementation	80
src/ obstacleAvoidance.cpp	81
src/ path.cpp	
Path class implementation	81
src/ pathFollower.cpp	81
src/ point.cpp	81
src/ prison.cpp	82
src/ pursuit.cpp	82
src/ pvector.cpp	82
src/ random.cpp	82
src/ scenario.cpp	82
src/ steeringBehavior.cpp	83
src/ wander.cpp	83
src/ windy.cpp	83
test/ test_suites.cpp	83

Chapter 5

Class Documentation

5.1 agent Class Reference

```
#include <agent.h>
```

Collaboration diagram for agent:

Public Member Functions

- [agent](#) (float x, float y)
- [agent](#) ()
- [~agent](#) ()
- void [updatePosition](#) (bool [arrive](#))
- void [setFeatures](#) (float s, float f, float r, float m)

Public Attributes

- string [name](#)
- color [fillColor](#)
- point [position](#)
- pvector [velocity](#)
- point [targetPoint](#)
- float [maxSpeed](#)
- float [maxForce](#)
- pvector [steering](#)
- pvector [force](#)
- pvector [acceleration](#)
- pvector [desiredVelocity](#)
- float [r](#)
- float [mass](#)
- int [id](#)
- bool [arrive](#) = false

5.1.1 Detailed Description

Definition at line 18 of file agent.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 agent() [1/2]

```
agent::agent (
    float x,
    float y )
```

Definition at line 11 of file agent.cpp.

```
11     {
12         position      = point(x, y);
13         velocity       = pvector(0.6, 0.0);
14         acceleration   = pvector(0.0, 0.0);
15         steering       = pvector(0.0, 0.0);
16         desiredVelocity = pvector(0.0, 0.0);
17         force          = pvector(0.0, 0.0);
18         targetPoint    = point(0.0, 0.0);
19         fillColor      = color(1.0, 0.0, 0.0);
20     }
```

5.1.2.2 agent() [2/2]

```
agent::agent ( )
```

Definition at line 9 of file agent.cpp.

```
9 {}
```

5.1.2.3 ~agent()

```
agent::~agent ( )
```

Definition at line 49 of file agent.cpp.

```
49 {}
```

5.1.3 Member Function Documentation

5.1.3.1 setFeatures()

```
void agent::setFeatures (
    float s,
    float f,
    float r,
    float m )
```

Definition at line 42 of file agent.cpp.

```
42     {
43         this->maxSpeed = s;
44         this->maxForce = f;
45         this->r = r;
46         this->mass = m;
47     }
```

5.1.3.2 updatePosition()

```
void agent::updatePosition (
    bool arrive )
```

Definition at line 22 of file agent.cpp.

```
22 {
23     force.limit(maxForce);
24     acceleration = force;
25     velocity += acceleration;
26
27     //arriving behavior implementation
28     if(arrive == true){
29         pvector diff = targetPoint - position;
30         if(diff.magnitude() > r)
31             velocity.limit(maxSpeed);
32         else
33             velocity.limit(maxSpeed * diff.magnitude() / r);
34     }
35     else
36         velocity.limit(maxSpeed);
37
38     position = position + velocity;
39     force = pvector(0,0);
40 }
```

Here is the call graph for this function:

5.1.4 Member Data Documentation

5.1.4.1 acceleration

```
pvector agent::acceleration
```

Definition at line 34 of file agent.h.

5.1.4.2 arrive

```
bool agent::arrive = false
```

Definition at line 39 of file agent.h.

5.1.4.3 desiredVelocity

```
pvector agent::desiredVelocity
```

Definition at line 35 of file agent.h.

5.1.4.4 fillColor

```
color agent::fillColor
```

Definition at line 26 of file agent.h.

5.1.4.5 force

```
pvector agent::force
```

Definition at line 33 of file agent.h.

5.1.4.6 id

```
int agent::id
```

Definition at line 38 of file agent.h.

5.1.4.7 mass

```
float agent::mass
```

Definition at line 37 of file agent.h.

5.1.4.8 maxForce

```
float agent::maxForce
```

Definition at line 31 of file agent.h.

5.1.4.9 maxSpeed

```
float agent::maxSpeed
```

Definition at line 30 of file agent.h.

5.1.4.10 name

```
string agent::name
```

Definition at line 25 of file agent.h.

5.1.4.11 position

```
point agent::position
```

Definition at line 27 of file agent.h.

5.1.4.12 r

```
float agent::r
```

Definition at line 36 of file agent.h.

5.1.4.13 steering

```
pvector agent::steering
```

Definition at line 32 of file agent.h.

5.1.4.14 targetPoint

```
point agent::targetPoint
```

Definition at line 29 of file agent.h.

5.1.4.15 velocity

```
pvector agent::velocity
```

Definition at line 28 of file agent.h.

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

- [include/agent.h](#)
- [src/agent.cpp](#)

5.2 color Class Reference

```
#include <color.h>
```

Collaboration diagram for color:

Public Member Functions

- [color](#) ()
default constructor.
- [color](#) (float r, float g, float b)
Constructor.
- void [createColors](#) ()
fills colors vector with 8 main colors in color bar
- [color](#) [getColor](#) (int i)
Constructor.

Public Attributes

- float [R](#)
red condiment
- float [G](#)
green condiment
- float [B](#)
blue condiment
- vector< [color](#) > [colors](#)
stores main colors

5.2.1 Detailed Description

Definition at line 20 of file color.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 [color\(\)](#) [1/2]

```
color::color ( )
```

default constructor.

Create a new color object.

See also

[color\(float r, float g, float b\)](#)

Definition at line 25 of file color.cpp.

```
26 {  
27  
28 }
```

5.2.2.2 color() [2/2]

```
color::color (
    float r,
    float g,
    float b )
```

Constructor.

Create a new color object.

Parameters

<i>r</i>	red (0-255)
<i>g</i>	green (0-255)
<i>b</i>	blue (0-255)

See also

[path\(\)](#)

Definition at line 13 of file color.cpp.

```
14 {
15     R = r;
16     G = g;
17     B = b;
18 }
```

5.2.3 Member Function Documentation

5.2.3.1 createColors()

```
void color::createColors ( )
```

fills colors vector with 8 main colors in color bar

creates main colors for future use

Definition at line 30 of file color.cpp.

```
31 {
32     colors.push_back(color(0.0, 0.0, 0.0));
33     colors.push_back(color(0.0, 0.0, 1.0));
34     colors.push_back(color(0.0, 1.0, 0.0));
35     colors.push_back(color(0.0, 1.0, 1.0));
36     colors.push_back(color(1.0, 0.0, 0.0));
37     colors.push_back(color(1.0, 0.0, 1.0));
38     colors.push_back(color(1.0, 1.0, 0.0));
39     colors.push_back(color(1.0, 1.0, 1.0));
40 }
```

5.2.3.2 getColor()

```
color color::getColor (
    int i )
```

Constructor.

returns specified color from colors vector

Parameters

<i>i</i>	gets specified color
----------	----------------------

Returns

requested pre-created color instance

Definition at line 20 of file color.cpp.

```
21 {  
22     return colors.at(i);  
23 }
```

Here is the caller graph for this function:

5.2.4 Member Data Documentation

5.2.4.1 B

```
float color::B
```

blue condiment

blue color ratio

Definition at line 69 of file color.h.

5.2.4.2 colors

```
vector<color> color::colors
```

stores main colors

vector of fundamental colors

Definition at line 75 of file color.h.

5.2.4.3 G

```
float color::G
```

green condiment

green color ratio

Definition at line 63 of file color.h.

5.2.4.4 R

```
float color::R
```

red condiment

red color ratio

Definition at line 57 of file color.h.

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

- [include/color.h](#)
- [src/color.cpp](#)

5.3 evade Class Reference

```
#include <evade.h>
```

Inheritance diagram for evade:

Collaboration diagram for evade:

Public Member Functions

- [evade](#) ()

Static Public Member Functions

- static void [loop](#) ()

Additional Inherited Members

5.3.1 Detailed Description

Definition at line 8 of file evade.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 evade()

```
evade::evade ( )
```

Definition at line 23 of file evade.cpp.

```
23     {
24     name = "evading";
25     createAgent(STATIC, nullptr, nullptr, nullptr);
26     callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
27 }
```

5.3.3 Member Function Documentation

5.3.3.1 loop()

```
void evade::loop ( ) [static]
```

Definition at line 8 of file evade.cpp.

```
8     {
9     for(auto it = agents.begin(); it < agents.end(); it++){
10         if((*it).name == "lion"){
11             (*it).targetPoint = view.getMousePosition();
12             (*it).force = behavior.seek(*it);
13             (*it).arrive = true;
14         }
15         else{//gazelle
16             (*it).force = behavior.evade(agents, *it, view);
17         }
18     }
19
20     refresh();
21 }
```

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

- include/[evade.h](#)
- src/[evade.cpp](#)

5.4 flee Class Reference

```
#include <flee.h>
```

Inheritance diagram for flee:

Collaboration diagram for flee:

Public Member Functions

- [flee](#) ()

Static Public Member Functions

- static void [loop](#) ()

Additional Inherited Members

5.4.1 Detailed Description

Definition at line 8 of file flee.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 flee()

```
flee::flee ( )
```

Definition at line 16 of file flee.cpp.

```
16     {
17         int agentCount = 196;
18         name = "fleeing troop";
19         createAgent(TROOP, &agentCount, nullptr, nullptr);
20         callback = reinterpret_cast<void(*)()> ( (void *)(&loop) );
21     }
```

5.4.3 Member Function Documentation

5.4.3.1 loop()

```
void flee::loop ( ) [static]
```

Definition at line 8 of file flee.cpp.

```
8     {
9         for(auto it = agents.begin(); it < agents.end(); it++){
10             (*it).force = behavior.flee((*it), view, view.getMousePosition());
11         }
12
13         refresh();
14     }
```

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

- [include/flee.h](#)
- [src/flee.cpp](#)

5.5 flock Class Reference

```
#include <flock.h>
```

Inheritance diagram for flock:

Collaboration diagram for flock:

Public Member Functions

- [flock](#) ()

Static Public Member Functions

- static void [loop](#) ()

Additional Inherited Members

5.5.1 Detailed Description

Definition at line 8 of file flock.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 flock()

`flock::flock ()`

Definition at line 28 of file flock.cpp.

```

28     {
29         int agentCount = 50;
30         float maxForce = 0.3;
31         float maxSpeed = 0.8;
32         name = "flocking agents";
33         createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
34         callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
35     }
```

5.5.3 Member Function Documentation

5.5.3.1 loop()

`void flock::loop () [static]`

Definition at line 8 of file flock.cpp.

```

8     {
9         for(auto it = agents.begin(); it < agents.end(); it++){
10             view.forceInScreen((*it));
11
12             pvector sep = behavior.separation(agents, *it);
13             sep.mul(1.5);
14             pvector ali = behavior.align(agents, *it);
15             ali.mul(4);
16             pvector coh = behavior.cohesion(agents, *it);
17             coh.mul(0.1);
18
19             (*it).force = sep + ali + coh;
20             (*it).desiredVelocity = (*it).force + (*it).velocity;
21             (*it).targetPoint = (*it).position + (*it).desiredVelocity;
22             (*it).arrive = true;
23         }
24         refresh();
25     }
26 }
```

Here is the call graph for this function:

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

- [include/flock.h](#)
- [src/flock.cpp](#)

5.6 flowField Class Reference

```
#include <flowField.h>
```

Collaboration diagram for flowField:

Public Member Functions

- [flowField](#) ()
default constructor.
- [flowField](#) ([pvector](#) p)
constructor.
- [pvector](#) [getField](#) (int x, int y)
get force for individual pixel

5.6.1 Detailed Description

Definition at line 18 of file flowField.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 flowField() [1/2]

```
flowField::flowField ( )
```

default constructor.

Create a new [flowField](#) object.

See also

[flowField\(pvector p\)](#)

Definition at line 15 of file flowField.cpp.

```
15 {}
```

5.6.2.2 flowField() [2/2]

```
flowField::flowField (  
    pvector p )
```

constructor.

Create a new [flowField](#) object.

Parameters

p	force vector
-----	--------------

See also

[flowField\(\)](#)

Definition at line 10 of file flowField.cpp.

```
11 {
12     createFlowField(p);
13 }
```

5.6.3 Member Function Documentation**5.6.3.1 getField()**

```
pvector flowField::getField (
    int x,
    int y )
```

get force for individual pixel

get force for a specific position

Parameters

x	x cprovidesoorinate
y	y coordinate

Returns

returns force at specified position

Definition at line 36 of file flowField.cpp.

```
37 {
38     return uniformField[x][y];
39 }
```

Here is the caller graph for this function:

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

- [include/flowField.h](#)
- [src/flowField.cpp](#)

5.7 graphics Class Reference

```
#include <graphics.h>
```

Collaboration diagram for graphics:

Public Member Functions

- void [drawWall](#) (float border, [color](#) color)
- void [drawAgent](#) ([agent](#) &agent, [color](#) &color)
- void [drawLine](#) ([point](#) p1, [point](#) p2, [color](#) cl)
- void [drawPath](#) ([path](#) &path, [color](#) color)
- void [drawPoint](#) ([point](#) p)
- void [drawCircle](#) ([point](#) p, float radius)
- void [drawText](#) (string text, [point](#) p)
- void [forceInScreen](#) ([agent](#) &agent)
- void [refreshScene](#) ()
- [point](#) [getMousePosition](#) ()
- void [initGraphics](#) (int *argv, char **argc, void(*callback)())

Static Public Member Functions

- static void [timerEvent](#) (int value)
- static void [handleKeypress](#) (unsigned char key, int x, int y)
- static void [mouseButton](#) (int button, int state, int x, int y)
- static void [handleResize](#) (int w, int h)
- static void [mouseMove](#) (int x, int y)

Static Public Attributes

- static int [target_x](#) = -WIDTH
- static int [target_y](#) = HEIGHT

5.7.1 Detailed Description

Definition at line 15 of file graphics.h.

5.7.2 Member Function Documentation

5.7.2.1 drawAgent()

```
void graphics::drawAgent (
    agent & agent,
    color & color )
```

Definition at line 162 of file graphics.cpp.

```
162                                     {
163     glPushMatrix();
164     glTranslatef(agent.position.x, agent.position.y, 0.0f);
165     glRotatef(agent.velocity.getAngle(), 0.0f, 0.0f, 1.0f);
166     glBegin(GL_TRIANGLES);
167     glColor3f( color.R, color.G, color.B);
168     glVertex3f( 1.0f, 0.0f, 0.0f);
169     glVertex3f(-1.0f, 0.5f, 0.0f);
170     glVertex3f(-1.0f, -0.5f, 0.0f);
171     glEnd();
172     glPopMatrix();
173 }
```

Here is the call graph for this function:

5.7.2.2 drawCircle()

```
void graphics::drawCircle (
    point p,
    float radius )
```

Definition at line 124 of file graphics.cpp.

```
124                                     {
125     glBegin(GL_LINE_STRIP);
126     glLineWidth(2);
127     for (int i = 0; i <= 300; i++) {
128         float angle = 2 * PI * i / 300;
129         float x = cos(angle) * radius;
130         float y = sin(angle) * radius;
131         glVertex2d(p.x + x, p.y + y);
132     }
133     glEnd();
134 }
```

5.7.2.3 drawLine()

```
void graphics::drawLine (
    point p1,
    point p2,
    color c1 )
```

Definition at line 115 of file graphics.cpp.

```
115                                     {
116     glColor3f( c1.R, c1.G, c1.B);
117     glLineWidth(2);
118     glBegin(GL_LINES);
119     glVertex2f(p1.x, p1.y);
120     glVertex2f(p2.x, p2.y);
121     glEnd();
122 }
```

5.7.2.4 drawPath()

```
void graphics::drawPath (
    path & path,
    color color )
```

Definition at line 102 of file graphics.cpp.

```
102                                     {
103     point p1, p2;
104     for(auto it = path.points.begin(); it < path.points.end()-1; it++){
105         p1 = point((*it).x, (*it).y - path.width/2) ;
106         p2 = point((*it+1).x, (*it+1).y - path.width/2);
107         drawLine(p1, p2, color.getColor(BLUE));
108     }
109     p1 = point((*it).x, (*it).y + path.width/2) ;
110     p2 = point((*it+1).x, (*it+1).y + path.width/2);
111     drawLine(p1, p2, color.getColor(BLUE));
112 }
113 }
```

Here is the call graph for this function:

5.7.2.5 drawPoint()

```
void graphics::drawPoint (
    point p )
```

Definition at line 136 of file graphics.cpp.

```
136     {
137         glColor3f(1,1,1);
138         glPointSize(4.0);
139         glBegin(GL_POINTS);
140         glVertex2f(p.x, p.y);
141         glEnd();
142     }
```

Here is the caller graph for this function:

5.7.2.6 drawText()

```
void graphics::drawText (
    string text,
    point p )
```

Definition at line 14 of file graphics.cpp.

```
14     {
15         glColor3f (0.0, 0.0, 1.0);
16         //glRasterPos2f(-34, 32.5);
17         glRasterPos2f(p.x, p.y);
18         for ( string::iterator it=text.begin(); it!=text.end(); ++it){
19             glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *it);
20         }
21     }
```

Here is the caller graph for this function:

5.7.2.7 drawWall()

```
void graphics::drawWall (
    float border,
    color color )
```

Definition at line 144 of file graphics.cpp.

```
144     {
145         point p1 {-border, border};
146         point p2 { border, border};
147         drawLine(p1, p2, color.getColor(BLUE));
148
149         p1 = point ( border, border);
150         p2 = point ( border, -border);
151         drawLine(p1, p2, color.getColor(BLUE));
152
153         p1 = point ( border, -border);
154         p2 = point ( -border, -border);
155         drawLine(p1, p2, color.getColor(BLUE));
156
157         p1 = point (-border, border);
158         p2 = point (-border, -border);
159         drawLine(p1, p2, color.getColor(BLUE));
160     }
```

Here is the call graph for this function:

5.7.2.8 forceInScreen()

```
void graphics::forceInScreen (
    agent & agent )
```

Definition at line 52 of file graphics.cpp.

```
52                                     {
53     if(agent.position.x > WIDTH)
54         agent.position.x -= 2 * WIDTH;
55     if(agent.position.x < -WIDTH)
56         agent.position.x += 2 * WIDTH;
57     if(agent.position.y > HEIGHT)
58         agent.position.y -= 2 * HEIGHT;
59     if(agent.position.y < -HEIGHT)
60         agent.position.y += 2 * HEIGHT;
61 }
```

5.7.2.9 getMousePosition()

```
point graphics::getMousePosition ( )
```

Definition at line 48 of file graphics.cpp.

```
48     {
49     return point (graphics::target_x, graphics::target_y);
50 }
```

Here is the call graph for this function:

5.7.2.10 handleKeypress()

```
void graphics::handleKeypress (
    unsigned char key,
    int x,
    int y ) [static]
```

Definition at line 99 of file graphics.cpp.

```
99                                     {
100     if (key == ESC){ exit(0); }
101 }
```

Here is the caller graph for this function:

5.7.2.11 handleResize()

```
void graphics::handleResize (
    int w,
    int h ) [static]
```

Definition at line 70 of file graphics.cpp.

```
70                                     {
71     glViewport(0, 0, w, h); //Tell OpenGL how to convert from coordinates to pixel values
72     glMatrixMode(GL_PROJECTION); //Switch to setting the camera perspective
73     glLoadIdentity(); //Reset the camera
74     //Set the camera perspective
75     gluPerspective(45.0, //The camera angle
76                    (double)w / (double)h, //The width-to-height ratio
77                    1.0, //The near z clipping coordinate
78                    200.0); //The far z clipping coordinate
79 }
```

Here is the caller graph for this function:

5.7.2.12 initGraphics()

```
void graphics::initGraphics (
    int * argv,
    char ** argc,
    void(*)() callback )
```

Definition at line 32 of file graphics.cpp.

```
32
33     glutInit(argv, argc);
34     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
35     glutInitWindowSize(400, 400);
36     glutCreateWindow("Autonomous Steering Agents");
37     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); //set background color
38     glEnable(GL_DEPTH_TEST);
39     glutDisplayFunc(*callback);
40     glutMouseFunc(graphics::mouseButton);
41     glutPassiveMotionFunc(graphics::mouseMove);
42     glutKeyboardFunc(graphics::handleKeypress);
43     glutReshapeFunc(graphics::handleResize);
44     glutTimerFunc(5, graphics::timerEvent, 0);
45     glutMainLoop();
46 }
```

Here is the call graph for this function:

5.7.2.13 mouseButton()

```
void graphics::mouseButton (
    int button,
    int state,
    int x,
    int y ) [static]
```

Definition at line 93 of file graphics.cpp.

```
93
94     if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN){
95         cout << "zdf";
96     }
97 }
```

Here is the caller graph for this function:

5.7.2.14 mouseMove()

```
void graphics::mouseMove (
    int x,
    int y ) [static]
```

Definition at line 63 of file graphics.cpp.

```
63
64     //TODO: mouse position to glut
65     //TODO: magic numbers
66     graphics::target_x = x / 5.88 - 34;
67     graphics::target_y = 34 - y / 5.88;
68 }
```

Here is the caller graph for this function:

5.7.2.15 refreshScene()

```
void graphics::refreshScene ( )
```

Definition at line 24 of file graphics.cpp.

```
24     {
25     glutSwapBuffers();
26     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
27     glMatrixMode(GL_MODELVIEW); //Switch to the drawing perspective
28     glLoadIdentity(); //Reset the drawing perspective
29     glTranslatef(0.0f, 0.0f, -85.0f); //Move to the center of the triangle
30 }
```

5.7.2.16 timerEvent()

```
void graphics::timerEvent (
    int value ) [static]
```

Definition at line 83 of file graphics.cpp.

```
83     {
84     glutPostRedisplay(); //Tell GLUT that the display has changed
85     glutTimerFunc(20, timerEvent, 0);
86     /*counter++;
87     if(counter == _100_MS * 2){
88         counter = 0;
89         graphics::timerEventFlag = true;
90     }*/
91 }
```

Here is the caller graph for this function:

5.7.3 Member Data Documentation

5.7.3.1 target_x

```
int graphics::target_x = -WIDTH [static]
```

Definition at line 33 of file graphics.h.

5.7.3.2 target_y

```
int graphics::target_y = HEIGHT [static]
```

Definition at line 34 of file graphics.h.

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

- [include/graphics.h](#)
- [src/graphics.cpp](#)

5.8 mouseFollower Class Reference

```
#include <mouseFollower.h>
```

Inheritance diagram for mouseFollower:

Collaboration diagram for mouseFollower:

Public Member Functions

- [mouseFollower](#) ()

Static Public Member Functions

- static void [loop](#) ()

Additional Inherited Members

5.8.1 Detailed Description

Definition at line 8 of file mouseFollower.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 mouseFollower()

```
mouseFollower::mouseFollower ( )
```

Definition at line 17 of file mouseFollower.cpp.

```
17         {
18     int agentCount = 30;
19     float maxForce = 0.3;
20     float maxSpeed = 0.6;
21     name = "mouse following";
22     createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
23     callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
24 }
```

5.8.3 Member Function Documentation

5.8.3.1 loop()

```
void mouseFollower::loop ( ) [static]
```

Definition at line 8 of file mouseFollower.cpp.

```
8      {
9      for(auto it = agents.begin(); it < agents.end(); it++){
10         (*it).targetPoint = view.getMousePosition();
11         (*it).force = behavior.seek(*it);
12         (*it).arrive = true;
13     }
14     refresh();
15 }
```

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

- include/mouseFollower.h
- src/mouseFollower.cpp

5.9 obstacle Class Reference

```
#include <obstacle.h>
```

Collaboration diagram for obstacle:

Public Member Functions

- [obstacle \(\)](#)
Default constructor.
- [obstacle \(point p, float r\)](#)
Constructor.

Public Attributes

- [point p](#)
x and y coordinates
- [float r](#)
the bigger radius the bigger the obstacle

5.9.1 Detailed Description

Definition at line 12 of file obstacle.h.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 obstacle() [1/2]

```
obstacle::obstacle ( )
```

Default constructor.

Create a new obstacle object.

See also

[obstacle\(point p, float r\);](#)

Definition at line 15 of file obstacle.cpp.

```
15 {}
```

5.9.2.2 obstacle() [2/2]

```
obstacle::obstacle (
    point p,
    float r )
```

Constructor.

Create a new obstacle object.

Parameters

<i>p</i>	center of the circular obstacle
<i>r</i>	radius of the obstacle

See also

[obstacle\(point p, float r\);](#)

Definition at line 17 of file obstacle.cpp.

```
17 {
18     this->p = p;
19     this->r = r;
20 }
```

5.9.3 Member Data Documentation**5.9.3.1 p**

```
point obstacle::p
```

x and y coordinates

center point of the obstacle

Definition at line 34 of file obstacle.h.

5.9.3.2 `r`

```
float obstacle::r
```

the bigger radius the bigger the obstacle

radius of the obstacle

Definition at line 40 of file obstacle.h.

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

- [include/obstacle.h](#)
- [src/obstacle.cpp](#)

5.10 obstacleAvoidance Class Reference

```
#include <obstacleAvoidance.h>
```

Inheritance diagram for obstacleAvoidance:

Collaboration diagram for obstacleAvoidance:

Public Member Functions

- [obstacleAvoidance](#) ()

Static Public Member Functions

- static void [loop](#) ()
- static void [createObstacle](#) (vector< [obstacle](#) > &[obstacles](#))

Static Public Attributes

- static vector< [obstacle](#) > [obstacles](#)

Additional Inherited Members

5.10.1 Detailed Description

Definition at line 9 of file obstacleAvoidance.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 obstacleAvoidance()

obstacleAvoidance::obstacleAvoidance ()

Definition at line 34 of file obstacleAvoidance.cpp.

```

34     {
35         name = "avoid obstacles";
36         createAgent(STATIC, nullptr, nullptr, nullptr);
37         createObstacle(obstacles);
38         callback = reinterpret_cast<void(*)()> ( (void *)(&loop) );
39     }
```

5.10.3 Member Function Documentation

5.10.3.1 createObstacle()

void obstacleAvoidance::createObstacle (
 vector< obstacle > & obstacles) [static]

Definition at line 28 of file obstacleAvoidance.cpp.

```

28     {
29         obstacles.push_back(obstacle(point(0,0), 8));
30         obstacles.push_back(obstacle(point(-20,0), 3));
31         obstacles.push_back(obstacle(point(20,-10), 4));
32     }
```

Here is the call graph for this function:

5.10.3.2 loop()

void obstacleAvoidance::loop () [static]

Definition at line 10 of file obstacleAvoidance.cpp.

```

10     {
11         for(auto it = agents.begin(); it < agents.end(); it++){
12             for(auto it = obstacles.begin(); it < obstacles.end(); it++){
13                 point p = (*it).p;
14                 view.drawCircle(p, (*it).r);
15             }
16
17             (*it).targetPoint = view.getMousePosition();
18             pvector seek = behavior.seek(*it);
19             seek.mul(0.5);
20
21             pvector avoid = behavior.avoid(obstacles, *it);
22             (*it).force = avoid + seek;
23             (*it).arrive = true;
24         }
25         refresh();
26     }
```

Here is the call graph for this function:

5.10.4 Member Data Documentation

5.10.4.1 obstacles

```
vector< obstacle > obstacleAvoidance::obstacles [static]
```

Definition at line 13 of file obstacleAvoidance.h.

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

- include/[obstacleAvoidance.h](#)
- src/[obstacleAvoidance.cpp](#)

5.11 path Class Reference

```
#include <path.h>
```

Collaboration diagram for path:

Public Member Functions

- [path](#) ()
Default constructor.
- [path](#) (float [width](#))
Constructor.
- void [addPoint](#) ([point](#) p)
adds a new point to the path

Public Attributes

- vector< [point](#) > [points](#)
points added to the path
- int [width](#)
defines width of the path

5.11.1 Detailed Description

Definition at line 15 of file path.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 path() [1/2]

```
path::path ( )
```

Default constructor.

Create a new path object.

See also

[path\(float width\)](#)

Definition at line 16 of file path.cpp.

```
17 {
18
19 }
```

5.11.2.2 path() [2/2]

```
path::path (
    float width )
```

Constructor.

Create a new path object.

Parameters

<i>width</i>	The width of the path.
--------------	------------------------

See also

[path\(\)](#)

Definition at line 21 of file path.cpp.

```
22 {
23     this->width = width;
24 }
```

5.11.3 Member Function Documentation**5.11.3.1 addPoint()**

```
void path::addPoint (
    point p )
```

adds a new point to the path

Used when customizing path

Parameters

<i>point</i>	new point to add to the path
--------------	------------------------------

Definition at line 11 of file path.cpp.

```
12 {  
13     points.push_back(p);  
14 }
```

Here is the caller graph for this function:

5.11.4 Member Data Documentation

5.11.4.1 [points](#)

```
vector<point> path::points
```

points added to the path

path is created from these points

Definition at line 43 of file path.h.

5.11.4.2 [width](#)

```
int path::width
```

defines width of the path

path width

Definition at line 49 of file path.h.

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

- [include/path.h](#)
- [src/path.cpp](#)

5.12 pathFollower Class Reference

```
#include <pathFollower.h>
```

Inheritance diagram for pathFollower:

Collaboration diagram for pathFollower:

Public Member Functions

- [pathFollower](#) ()

Static Public Member Functions

- static void [loop](#) ()
- static void [createPath](#) ([path](#) &p)

Static Public Attributes

- static [path](#) [myPath](#)

Additional Inherited Members

5.12.1 Detailed Description

Definition at line 8 of file pathFollower.h.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 pathFollower()

```
pathFollower::pathFollower ( )
```

Definition at line 28 of file pathFollower.cpp.

```
28 {
29     int agentCount = 40;
30     float maxForce = 0.2;
31     float maxSpeed = 0.4;
32     myPath = path(8);
33     createPath(myPath);
34     name = "path following";
35     createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
36     callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
37 }
```

5.12.3 Member Function Documentation

5.12.3.1 createPath()

```
void pathFollower::createPath (
    path & p ) [static]
```

Definition at line 21 of file pathFollower.cpp.

```
21 {
22     p.addPoint(point(-40, 5));
23     p.addPoint(point(-14, 15));
24     p.addPoint(point( 10, 7));
25     p.addPoint(point( 40, 12));
26 }
```

Here is the call graph for this function:

5.12.3.2 loop()

```
void pathFollower::loop ( ) [static]
```

Definition at line 10 of file pathFollower.cpp.

```
10 {
11     for(auto it = agents.begin(); it < agents.end(); it++){
12         view.drawPath(myPath, myColor);
13         pvector seek = behavior.stayInPath_2(*it, myPath, view);
14         pvector sep = behavior.separation(agents, *it);
15         sep.mul(5);
16         (*it).force = sep + seek;
17     }
18     refresh();
19 }
```

Here is the call graph for this function:

5.12.4 Member Data Documentation

5.12.4.1 myPath

```
path pathFollower::myPath [static]
```

Definition at line 12 of file pathFollower.h.

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

- include/[pathFollower.h](#)
- src/[pathFollower.cpp](#)

5.13 point Class Reference

```
#include <point.h>
```

Collaboration diagram for point:

Public Member Functions

- [point](#) (float [x](#), float [y](#))
- [point](#) ()
- void [div](#) (float [d](#))
- void [mul](#) (float [d](#))
- void [print](#) (const string &[s](#))
- [point operator+](#) ([pvector](#) const &[obj](#))
- [point operator+](#) ([point](#) const &[obj](#))
- [pvector operator-](#) ([point](#) const &[obj](#))
- bool [operator==](#) ([point](#) const &[obj](#))
- void [getNormalPoint](#) ([point](#) [predicted](#), [point](#) [start](#), [point](#) [end](#))

Public Attributes

- float [x](#)
- float [y](#)

5.13.1 Detailed Description

Definition at line 8 of file [point.h](#).

5.13.2 Constructor & Destructor Documentation

5.13.2.1 [point\(\)](#) [1/2]

```
point::point (
    float x,
    float y )
```

Definition at line 8 of file [point.cpp](#).

```
8      {
9      this->x = x;
10     this->y = y;
11 }
```

5.13.2.2 [point\(\)](#) [2/2]

```
point::point ( )
```

Definition at line 13 of file [point.cpp](#).

```
13 {}
```

Here is the caller graph for this function:

5.13.3 Member Function Documentation

5.13.3.1 div()

```
void point::div (
    float d )
```

Definition at line 28 of file point.cpp.

```
28     {
29         x = x / d;
30         y = y / d;
31     }
```

Here is the caller graph for this function:

5.13.3.2 getNormalPoint()

```
void point::getNormalPoint (
    point predicted,
    point start,
    point end )
```

Definition at line 53 of file point.cpp.

```
53     {
54         pvector a = predicted - start;
55         pvector b = end - start;
56         b.normalize();
57         float a_dot_b = a.dotProduct(b);
58         b.mul(a_dot_b);
59         point normalPoint = start + b;
60         this->x = normalPoint.x;
61         this->y = normalPoint.y;
62     }
```

Here is the call graph for this function: Here is the caller graph for this function:

5.13.3.3 mul()

```
void point::mul (
    float d )
```

Definition at line 33 of file point.cpp.

```
33     {
34         x = x * d;
35         y = y * d;
36     }
```

Here is the caller graph for this function:

5.13.3.4 operator+() [1/2]

```
point point::operator+ (
    point const & obj )
```

Definition at line 39 of file point.cpp.

```
39     {
40         point res;
41         res.x = x + obj.x;
42         res.y = y + obj.y;
43         return res;
44     }
```

5.13.3.5 operator+() [2/2]

```
point point::operator+ (
    pvector const & obj )
```

Definition at line 15 of file point.cpp.

```
15 {
16     point res;
17     res.x = x + obj.x;
18     res.y = y + obj.y;
19     return res;
20 }
```

5.13.3.6 operator-()

```
pvector point::operator- (
    point const & obj )
```

Definition at line 46 of file point.cpp.

```
46 {
47     pvector res;
48     res.x = x - obj.x;
49     res.y = y - obj.y;
50     return res;
51 }
```

5.13.3.7 operator==()

```
bool point::operator== (
    point const & obj )
```

Definition at line 22 of file point.cpp.

```
22 {
23     if(x == obj.x && y == obj.y)
24         return true;
25     return false;
26 }
```

5.13.3.8 print()

```
void point::print (
    const string & s )
```

Definition at line 64 of file point.cpp.

```
64 {
65     cout << " " << s << " " << x << " " << y << endl;
66 }
```

5.13.4 Member Data Documentation

5.13.4.1 x

```
float point::x
```

Definition at line 10 of file point.h.

5.13.4.2 y

```
float point::y
```

Definition at line 11 of file point.h.

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

- [include/point.h](#)
- [src/point.cpp](#)

5.14 prison Class Reference

```
#include <prison.h>
```

Inheritance diagram for prison:

Collaboration diagram for prison:

Public Member Functions

- [prison](#) ()

Static Public Member Functions

- static void [loop](#) ()

Additional Inherited Members

5.14.1 Detailed Description

Definition at line 8 of file prison.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 prison()

```
prison::prison ( )
```

Definition at line 17 of file prison.cpp.

```
17     {
18     int agentCount = 30;
19     float maxForce = 0.5;
20     float maxSpeed = 0.6;
21
22     name = "stay in prison";
23     createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
24     callback = reinterpret_cast<void(*)()> ( (void *)(&loop) );
25 }
```

5.14.3 Member Function Documentation

5.14.3.1 loop()

```
void prison::loop ( ) [static]
```

Definition at line 8 of file prison.cpp.

```
8     {
9     for(auto it = agents.begin(); it < agents.end(); it++){
10         view.drawWall(WALL, myColor);
11         (*it).force = behavior.stayInArea(*it, WALL - DISTANCE);
12         (*it).force += behavior.separation(agents, *it);
13     }
14     refresh();
15 }
```

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

- [include/prison.h](#)
- [src/prison.cpp](#)

5.15 pursuit Class Reference

```
#include <pursuit.h>
```

Inheritance diagram for pursuit:

Collaboration diagram for pursuit:

Public Member Functions

- [pursuit \(\)](#)

Static Public Member Functions

- static void [loop \(\)](#)

Additional Inherited Members

5.15.1 Detailed Description

Definition at line 8 of file `pursuit.h`.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 `pursuit()`

```
pursuit::pursuit ( )
```

Definition at line 23 of file `pursuit.cpp`.

```
23     {
24         name = "pursuit";
25         createAgent(STATIC, nullptr, nullptr, nullptr);
26         callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
27     }
```

5.15.3 Member Function Documentation

5.15.3.1 `loop()`

```
void pursuit::loop ( ) [static]
```

Definition at line 8 of file `pursuit.cpp`.

```
8     {
9         for(auto it = agents.begin(); it < agents.end(); it++){
10             if((*it).name == "gazelle"){
11                 (*it).targetPoint = view.getMousePosition();
12                 (*it).force = behavior.seek(*it);
13             }
14             else{//lion
15                 (*it).force = behavior.pursuit(agents, *it, view);
16             }
17             (*it).arrive = true;
18         }
19         refresh();
20     }
21 }
```

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

- `include/pursuit.h`
- `src/pursuit.cpp`

5.16 pvector Class Reference

```
#include <pvector.h>
```

Collaboration diagram for `pvector`:

Public Member Functions

- [pvector](#) ()
- [pvector](#) (float [x](#), float [y](#))
- float [magnitude](#) ()
- [pvector](#) & [normalize](#) ()
- void [div](#) (float [i](#))
- void [mul](#) (float [i](#))
- void [add](#) ([pvector](#) [p](#))
- void [limit](#) (float [limit](#))
- float [getAngle](#) ()
- float [dotProduct](#) ([pvector](#) [v](#))
- float [angleBetween](#) ([pvector](#) [v](#))
- [pvector](#) operator+= ([pvector](#) const &[obj](#))
- [pvector](#) operator+ ([pvector](#) const &[obj](#))
- [pvector](#) operator- ([pvector](#) const &[obj](#))
- [pvector](#) operator- ([point](#) const &[obj](#))
- [pvector](#) operator+ ([point](#) const &[obj](#))
- bool operator== ([pvector](#) const &[obj](#))
- void [print](#) (const string &[s](#))

Public Attributes

- float [x](#)
- float [y](#)

5.16.1 Detailed Description

Definition at line 11 of file [pvector.h](#).

5.16.2 Constructor & Destructor Documentation

5.16.2.1 [pvector\(\)](#) [1/2]

```
pvector::pvector ( )
```

Definition at line 25 of file [pvector.cpp](#).
25 {}

5.16.2.2 [pvector\(\)](#) [2/2]

```
pvector::pvector (
    float x,
    float y )
```

Definition at line 27 of file [pvector.cpp](#).
27 {
28 this->[x](#) = [x](#);
29 this->[y](#) = [y](#);
30 }

5.16.3 Member Function Documentation

5.16.3.1 add()

```
void pvector::add (
    pvector p )
```

Definition at line 42 of file pvector.cpp.

```
42     {
43         x = x + p.x;
44         y = y + p.y;
45     }
```

5.16.3.2 angleBetween()

```
float pvector::angleBetween (
    pvector v )
```

Definition at line 15 of file pvector.cpp.

```
15     {
16         float angle = this->dotProduct(v) / (this->magnitude() * v.magnitude());
17         angle = acos(angle) * 180 / PI;
18         return angle;
19     }
```

Here is the call graph for this function: Here is the caller graph for this function:

5.16.3.3 div()

```
void pvector::div (
    float i )
```

Definition at line 32 of file pvector.cpp.

```
32     {
33         x = x / i;
34         y = y / i;
35     }
```

Here is the caller graph for this function:

5.16.3.4 dotProduct()

```
float pvector::dotProduct (
    pvector v )
```

Definition at line 21 of file pvector.cpp.

```
21     {
22         return ((x * v.x) + (y * v.y));
23     }
```

Here is the caller graph for this function:

5.16.3.5 getAngle()

```
float pvector::getAngle ( )
```

Definition at line 9 of file pvector.cpp.

```
9      {  
10     float angle;  
11     angle = atan2 (this->y, this->x) * 180 / PI;  
12     return angle;  
13 }
```

Here is the caller graph for this function:

5.16.3.6 limit()

```
void pvector::limit (  
    float limit )
```

Definition at line 64 of file pvector.cpp.

```
64      {  
65     this->normalize();  
66     this->mul(limit);  
67 }
```

Here is the call graph for this function: Here is the caller graph for this function:

5.16.3.7 magnitude()

```
float pvector::magnitude ( )
```

Definition at line 47 of file pvector.cpp.

```
47      {  
48     return sqrt((this->x * this->x) + (this->y * this->y));  
49 }
```

Here is the caller graph for this function:

5.16.3.8 mul()

```
void pvector::mul (  
    float i )
```

Definition at line 37 of file pvector.cpp.

```
37      {  
38     x = x * i;  
39     y = y * i;  
40 }
```

Here is the caller graph for this function:

5.16.3.9 normalize()

```
pvector & pvector::normalize ( )
```

Definition at line 51 of file pvector.cpp.

```
51     {
52     float magnitude = this->magnitude();
53     if(magnitude != 0){
54         this->x = this->x / magnitude;
55         this->y = this->y / magnitude;
56     }
57     else{
58         this->x = 0;
59         this->y = 0;
60     }
61     return *this;
62 }
```

Here is the caller graph for this function:

5.16.3.10 operator+() [1/2]

```
pvector pvector::operator+ (
    point const & obj )
```

Definition at line 88 of file pvector.cpp.

```
88     {
89     pvector res;
90     res.x = x + obj.x;
91     res.y = y + obj.y;
92     return res;
93 }
```

5.16.3.11 operator+() [2/2]

```
pvector pvector::operator+ (
    pvector const & obj )
```

Definition at line 69 of file pvector.cpp.

```
69     {
70     pvector res;
71     res.x = x + obj.x;
72     res.y = y + obj.y;
73     return res;
74 }
```

5.16.3.12 operator+=()

```
pvector pvector::operator+= (
    pvector const & obj )
```

Definition at line 76 of file pvector.cpp.

```
76     {
77     x = x + obj.x;
78     y = y + obj.y;
79     return *this;
80 }
```

5.16.3.13 operator-() [1/2]

```
pvector pvector::operator- (
    point const & obj )
```

Definition at line 95 of file pvector.cpp.

```
95 {
96     pvector res;
97     res.x = x - obj.x;
98     res.y = y - obj.y;
99     return res;
100 }
```

5.16.3.14 operator-() [2/2]

```
pvector pvector::operator- (
    pvector const & obj )
```

Definition at line 106 of file pvector.cpp.

```
106 {
107     pvector res;
108     res.x = x - obj.x;
109     res.y = y - obj.y;
110     return res;
111 }
```

5.16.3.15 operator==()

```
bool pvector::operator== (
    pvector const & obj )
```

Definition at line 82 of file pvector.cpp.

```
82 {
83     if(x == obj.x && y == obj.y)
84         return true;
85     return false;
86 }
```

5.16.3.16 print()

```
void pvector::print (
    const string & s )
```

Definition at line 102 of file pvector.cpp.

```
102 {
103     cout << s << " " << x << " " << y << endl;
104 }
```

5.16.4 Member Data Documentation

5.16.4.1 x

```
float pvector::x
```

Definition at line 13 of file pvector.h.

5.16.4.2 y

```
float pvector::y
```

Definition at line 14 of file pvector.h.

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

- [include/pvector.h](#)
- [src/pvector.cpp](#)

5.17 random Class Reference

```
#include <random.h>
```

Collaboration diagram for random:

Static Public Member Functions

- static void [createRandomArray](#) (int *arr, int size)

5.17.1 Detailed Description

Definition at line 3 of file random.h.

5.17.2 Member Function Documentation

5.17.2.1 createRandomArray()

```
void random::createRandomArray (
    int * arr,
    int size ) [static]
```

Definition at line 7 of file random.cpp.

```
7
8     srand(time(NULL));
9     for(int i=0; i<size; i++)
10         arr[i] = i+1;
11
12     for (int i=0; i < size; i++){
13         int r = rand() % size;
14         swap(arr[i], arr[r]);
15     }
16 }
```

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

- include/[random.h](#)
- src/[random.cpp](#)

5.18 scenario Class Reference

```
#include <scenario.h>
```

Inheritance diagram for scenario:

Collaboration diagram for scenario:

Public Member Functions

- [scenario](#) ()
- void [createAgent](#) (int type, int *count, float *force, float *speed)
- void [initGL](#) (int *argv, char **argc)

Static Public Member Functions

- static void [refresh](#) ()

Public Attributes

- void(* [callback](#))()

Static Public Attributes

- static vector< [agent](#) > [agents](#)
- static [graphics view](#)
- static [steeringBehavior](#) [behavior](#)
- static [color](#) [myColor](#)
- static string [name](#)

5.18.1 Detailed Description

Definition at line 12 of file scenario.h.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 scenario()

```
scenario::scenario ( )
```

Definition at line 18 of file scenario.cpp.

```
19 {  
20     srand(time(NULL));  
21     myColor.createColors();  
22     view = graphics();  
23 }
```

5.18.3 Member Function Documentation

5.18.3.1 createAgent()

```
void scenario::createAgent (  
    int type,  
    int * count,  
    float * force,  
    float * speed )
```

Definition at line 95 of file scenario.cpp.

```
96 {  
97     if(type == TROOP){  
98         createTroop(*count);  
99     }  
100     else if(type == RANDOM){  
101         createRandomAgents(*count, *force, *speed);  
102     }  
103     else if(type == STATIC){  
104         createStaticAgents();  
105     }  
106     else{  
107         //error message  
108     }  
109 }
```

5.18.3.2 initGL()

```
void scenario::initGL (  
    int * argv,  
    char ** argc )
```

Definition at line 13 of file scenario.cpp.

```
13 {  
14     view.initGraphics(argc, argv, callback);  
15 }
```

Here is the caller graph for this function:

5.18.3.3 refresh()

```
void scenario::refresh ( ) [static]
```

Definition at line 25 of file scenario.cpp.

```
25         {
26     for(auto it = agents.begin(); it < agents.end(); it++){
27         (*it).updatePosition((*it).arrive);
28         view.drawAgent(*it, (*it).fillColor);
29     }
30
31     view.drawText(name, point(-34, 32.25)); //TODO: magic numbers, define left corner
32     view.refreshScene();
33 }
```

Here is the call graph for this function:

5.18.4 Member Data Documentation

5.18.4.1 agents

```
vector< agent > scenario::agents [static]
```

Definition at line 18 of file scenario.h.

5.18.4.2 behavior

```
steeringBehavior scenario::behavior [static]
```

Definition at line 20 of file scenario.h.

5.18.4.3 callback

```
void(* scenario::callback) ()
```

Definition at line 23 of file scenario.h.

5.18.4.4 myColor

```
color scenario::myColor [static]
```

Definition at line 21 of file scenario.h.

5.18.4.5 name

```
string scenario::name [static]
```

Definition at line 22 of file scenario.h.

5.18.4.6 view

```
graphics scenario::view [static]
```

Definition at line 19 of file scenario.h.

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

- include/scenario.h
- src/scenario.cpp

5.19 steeringBehavior Class Reference

```
#include <steeringBehavior.h>
```

Collaboration diagram for steeringBehavior:

Public Member Functions

- [pvector stayInArea](#) ([agent &agent](#), int turnPoint)
- [pvector inFlowField](#) ([agent &agent](#), [flowField](#) &flow)
- [pvector stayInPath](#) ([agent &agent](#), [path](#) &path)
- [pvector stayInPath_2](#) ([agent &agent](#), [path](#) &path, [graphics](#) view)
- [pvector seek](#) ([agent &agent](#))
- [pvector separation](#) (vector< [agent](#) > agents, [agent &agent](#))
- [pvector cohesion](#) (vector< [agent](#) > boids, [agent &agent](#))
- [pvector align](#) (vector< [agent](#) > boids, [agent &agent](#))
- [pvector wander](#) ([agent &agent](#))
- [pvector pursuit](#) (vector< [agent](#) > boids, [agent](#) &pursuer, [graphics](#) view)
- [pvector evade](#) (vector< [agent](#) > boids, [agent](#) &evader, [graphics](#) view)
- [pvector flee](#) ([agent &agent](#), [graphics](#) &view, [point](#) p)
- [pvector avoid](#) (vector< [obstacle](#) > obstacles, [agent &agent](#))
- void [setAngle](#) ([pvector](#) &p, float angle)

5.19.1 Detailed Description

Definition at line 28 of file steeringBehavior.h.

5.19.2 Member Function Documentation

5.19.2.1 align()

```
pvector steeringBehavior::align (
    vector< agent > boids,
    agent & agent )
```

Definition at line 105 of file steeringBehavior.cpp.

```
105 {
106     float neighborDist = 30; //TODO: magic number
107     pvector sum {0,0};
108     int count = 0;
109     for(auto it = boids.begin(); it < boids.end(); it++){
110         float d = (agent.position - (*it).position).magnitude();
111         if( (d > 0) && (d < neighborDist) ){
112             sum += (*it).velocity;
113             count++;
114         }
115     }
116     if(count > 0){
117         sum.div(count);
118         sum.normalize().mul(agent.maxSpeed);
119         agent.steering = sum - agent.velocity;
120         return agent.steering;
121     }
122     return pvector(0,0);
123 }
```

Here is the call graph for this function:

5.19.2.2 avoid()

```
pvector steeringBehavior::avoid (
    vector< obstacle > obstacles,
    agent & agent )
```

Definition at line 166 of file steeringBehavior.cpp.

```
166 {
167     float dynamic_length = agent.velocity.magnitude() / agent.maxSpeed;
168     pvector vel = agent.velocity;
169     vel.normalize().mul(dynamic_length);
170     pvector ahead = vel + agent.position;
171     vel.mul(6);
172     pvector ahead2 = vel + agent.position;
173     //view.drawPoint(point(ahead.x, ahead.y));
174     //view.drawPoint(point(ahead2.x, ahead2.y));
175
176     for(auto it = obstacles.begin(); it < obstacles.end(); it++){
177         float dist = (ahead - (*it).p).magnitude();
178         float dist2 = (ahead2 - (*it).p).magnitude();
179         if(dist < (*it).r + 2 || dist2 < (*it).r + 2){
180             pvector avoidance = ahead - (*it).p;
181             avoidance.normalize().mul(20);
182             /*a = point(avoidance.x, avoidance.y);
183             view.drawLine(agent.position, agent.position + a, color(0,1,0));*/
184             return avoidance;
185         }
186     }
187     return pvector(0,0);
188 }
```

Here is the call graph for this function:

5.19.2.3 cohesion()

```
pvector steeringBehavior::cohesion (
    vector< agent > boids,
    agent & agent )
```

Definition at line 125 of file steeringBehavior.cpp.

```
125 {
126     float neighborDist = 20; //TODO: magic number
127     point sum {0,0};
128     int count = 0;
129     for(auto it = boids.begin(); it < boids.end(); it++){
130         float d = (agent.position - (*it).position).magnitude();
131         if( (d > 0) && (d < neighborDist) ){
132             sum = sum + (*it).position;
133             count++;
134         }
135     }
136     if(count > 0){
137         sum.div(count);
138         agent.targetPoint = sum;
139         return seek(agent);
140     }
141     return pvector(0,0);
142 }
```

Here is the call graph for this function:

5.19.2.4 evade()

```
pvector steeringBehavior::evade (
    vector< agent > boids,
    agent & evader,
    graphics view )
```

Definition at line 36 of file steeringBehavior.cpp.

```
36 {
37     agent target;
38     for(auto it = boids.begin(); it < boids.end(); it++){
39         if((*it).name == "lion"){
40             target = *it;
41         }
42     }
43
44     point p = point(evader.position.x + 2, evader.position.y - 2);
45     view.drawText(evader.name, p);
46     p = point(target.position.x + 2, target.position.y - 2);
47     view.drawText(target.name, p);
48
49     pvector targetVel = target.velocity;
50     targetVel.mul(5); //TODO: magic number
51
52     point futurePos = target.position + targetVel;
53     view.drawPoint(futurePos);
54
55     pvector dist = evader.position - futurePos;
56     dist.normalize().mul( 1 / dist.magnitude() );
57
58     evader.targetPoint = evader.position + dist;
59     return flee(evader, view, futurePos);
60 }
```

Here is the call graph for this function:

5.19.2.5 flee()

```
pvector steeringBehavior::flee (
    agent & agent,
    graphics & view,
    point p )
```

Definition at line 20 of file steeringBehavior.cpp.

```
20 {
21     pvector dist = agent.targetPoint - p;
22     view.drawPoint(agent.targetPoint);
23
24     if(dist.magnitude() < 15){ //TODO: magic number
25         agent.arrive = false;
26         agent.desiredVelocity = agent.position - p;
27     }
28     else{
29         agent.arrive = true;
30         agent.desiredVelocity = agent.targetPoint - agent.position;
31     }
32     agent.steering = agent.desiredVelocity - agent.velocity;
33     return agent.steering;
34 }
```

Here is the call graph for this function:

5.19.2.6 inFlowField()

```
pvector steeringBehavior::inFlowField (
    agent & agent,
    flowField & flow )
```

Definition at line 236 of file steeringBehavior.cpp.

```
236 {
237     //pos_x, pos_y must be non negative integer
238     int pos_x = abs((int)agent.position.x) % WIDTH;
239     int pos_y = abs((int)agent.position.y) % HEIGHT;
240     //TODO: modification required for non uniform fields
241     return flow.getField(pos_x, pos_y);
242 }
```

Here is the call graph for this function:

5.19.2.7 pursuit()

```
pvector steeringBehavior::pursuit (
    vector< agent > boids,
    agent & pursuer,
    graphics view )
```

Definition at line 62 of file steeringBehavior.cpp.

```
62 {
63     agent target;
64     for(auto it = boids.begin(); it < boids.end(); it++){
65         if((*it).name == "gazelle"){
66             target = *it;
67         }
68     }
69
70     point p = point(target.position.x + 2, target.position.y - 2);
71     view.drawText(target.name, p);
72     p = point(pursuer.position.x + 2, pursuer.position.y - 2);
73     view.drawText(pursuer.name, p);
74
75     float dist = (target.position - pursuer.position).magnitude();
76     float t = dist / target.maxSpeed;
77
78     pvector targetVel = target.velocity;
79     targetVel.mul(t);
80     point futurePos = target.position + targetVel;
81     pursuer.targetPoint = futurePos;
82     return seek(pursuer);
83 }
```

Here is the call graph for this function:

5.19.2.8 seek()

```
pvector steeringBehavior::seek (
    agent & agent )
```

Definition at line 190 of file steeringBehavior.cpp.

```
190 {
191     agent.desiredVelocity = agent.targetPoint - agent.position;
192     agent.steering = agent.desiredVelocity - agent.velocity;
193     return agent.steering;
194 }
```

5.19.2.9 separation()

```
pvector steeringBehavior::separation (
    vector< agent > agents,
    agent & agent )
```

Definition at line 144 of file steeringBehavior.cpp.

```
144 {
145     float desiredSeparation = 5; //TODO: magic number
146     pvector sum = pvector(0,0);
147     int count = 0;
148     for(auto it = agents.begin(); it < agents.end(); it++){
149         float d = (agent.position - (*it).position).magnitude();
150         if( (d > 0) && (d < desiredSeparation) ){
151             pvector diff = agent.position - (*it).position;
152             diff.normalize().div(d);
153             sum = sum + diff;
154             count++;
155         }
156     }
157     if(count > 0){
158         sum.div(count);
159         sum.normalize().mul(agent.maxSpeed);
160         agent.steering = sum - agent.velocity;
161         return agent.steering;
162     }
163     return pvector(0,0);
164 }
```

Here is the call graph for this function:

5.19.2.10 setAngle()

```
void steeringBehavior::setAngle (
    pvector & p,
    float angle )
```

Definition at line 15 of file steeringBehavior.cpp.

```
15 {
16     p.x = cos ( angle * PI / 180.0 );
17     p.y = sin ( angle * PI / 180.0 );
18 }
```

5.19.2.11 stayInArea()

```
pvector steeringBehavior::stayInArea (
    agent & agent,
    int turnPoint )
```

Definition at line 244 of file steeringBehavior.cpp.

```
244                                     {
245     if(agent.position.x >= turnPoint){
246         agent.desiredVelocity = pvector( -agent.maxSpeed, agent.velocity.y );
247         agent.steering = agent.desiredVelocity - agent.velocity;
248         return agent.steering;
249     }
250     else if(agent.position.x <= -turnPoint){
251         agent.desiredVelocity = pvector( agent.maxSpeed, agent.velocity.y );
252         agent.steering = agent.desiredVelocity - agent.velocity;
253         return agent.steering;
254     }
255     else if(agent.position.y >= turnPoint){
256         agent.desiredVelocity = pvector( agent.velocity.x, -agent.maxSpeed );
257         agent.steering = agent.desiredVelocity - agent.velocity;
258         return agent.steering;
259     }
260     else if(agent.position.y <= -turnPoint){
261         agent.desiredVelocity = pvector( agent.velocity.x, agent.maxSpeed );
262         agent.steering = agent.desiredVelocity - agent.velocity;
263         return agent.steering;
264     }
265     return pvector(0,0);
266 }
```

5.19.2.12 stayInPath()

```
pvector steeringBehavior::stayInPath (
    agent & agent,
    path & path )
```

Definition at line 218 of file steeringBehavior.cpp.

```
218                                     {
219     point start = path.points.at(0);
220     point end = path.points.at(1);
221     point predictedPos = agent.position + agent.velocity;
222     point normalPoint;
223     normalPoint.getNormalPoint(predictedPos, start, end);
224     pvector b = end - start;
225     b.normalize();
226
227     pvector distance = predictedPos - normalPoint;
228     agent.targetPoint = normalPoint + b;
229     //view.drawLine(predictedPos, normalPoint);
230     //view.drawPoint(targetPoint);
231     if(distance.magnitude() > path.width / 8)
232         return seek(agent);
233     return pvector(0,0);
234 }
```

Here is the call graph for this function:

5.19.2.13 stayInPath_2()

```
pvector steeringBehavior::stayInPath_2 (
    agent & agent,
    path & path,
    graphics view )
```

Definition at line 196 of file steeringBehavior.cpp.

```

196
197     float worldRecord = 1000000; //TODO: magic number
198     point normalPoint, predictedPos, start, end;
199     pvector distance;
200     for(auto it = path.points.begin(); it < path.points.end()-1; it++){
201         start = point((*it).x, (*it).y);
202         end = point((*it+1).x, (*it+1).y);
203         predictedPos = agent.position + agent.velocity;
204         normalPoint.getNormalPoint(predictedPos, start, end);
205         if (normalPoint.x < start.x || normalPoint.x > end.x){
206             normalPoint = end;
207         }
208         distance = predictedPos - normalPoint;
209         if (distance.magnitude() < worldRecord){
210             worldRecord = distance.magnitude();
211             agent.targetPoint = end;
212         }
213         view.drawPoint(agent.targetPoint);
214     }
215     return seek(agent);
216 }

```

Here is the call graph for this function:

5.19.2.14 wander()

```

pvector steeringBehavior::wander (
    agent & agent )

```

Definition at line 85 of file steeringBehavior.cpp.

```

85
86     pvector circleCenter = agent.velocity;
87     circleCenter.normalize().mul(CIRCLE_DISTANCE + CIRCLE_RADIUS);
88
89     int wanderAngle = (rand() % 360);
90     pvector displacement {0, 1};
91     setAngle(displacement, wanderAngle);
92     displacement.mul(CIRCLE_RADIUS);
93
94     agent.desiredVelocity = displacement + circleCenter;
95     agent.steering = agent.desiredVelocity - agent.velocity;
96
97     //move it to the center when it is out of screen
98     if(agent.position.x > WIDTH || agent.position.x < -WIDTH ||
99        agent.position.y > HEIGHT || agent.position.y < -HEIGHT)
100         agent.position = point(0,0);
101
102     return agent.steering;
103 }

```

Here is the call graph for this function:

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

- include/steeringBehavior.h
- src/steeringBehavior.cpp

5.20 wander Class Reference

```
#include <wander.h>
```

Inheritance diagram for wander:

Collaboration diagram for wander:

Public Member Functions

- [wander\(\)](#)

Static Public Member Functions

- static void [loop\(\)](#)

Additional Inherited Members

5.20.1 Detailed Description

Definition at line 8 of file wander.h.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 wander()

```
wander::wander ( )
```

Definition at line 16 of file wander.cpp.

```
16     {
17         int agentCount = 30;
18         float maxForce = 0.3;
19         float maxSpeed = 0.6;
20
21         name = "wandering objects";
22         createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
23         callback = reinterpret_cast<void(*)()> ( (void *)(&loop) );
24     }
```

5.20.3 Member Function Documentation

5.20.3.1 loop()

```
void wander::loop ( ) [static]
```

Definition at line 8 of file wander.cpp.

```
8     {
9         for(auto it = agents.begin(); it < agents.end(); it++){
10             (*it).force = behavior.wander(*it);
11         }
12
13         refresh();
14     }
```

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

- include/[wander.h](#)
- src/[wander.cpp](#)

5.21 windy Class Reference

```
#include <windy.h>
```

Inheritance diagram for windy:

Collaboration diagram for windy:

Public Member Functions

- [windy\(\)](#)

Static Public Member Functions

- static void [loop\(\)](#)

Static Public Attributes

- static [flowField](#) [flow](#)

Additional Inherited Members

5.21.1 Detailed Description

Definition at line 9 of file windy.h.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 windy()

```
windy::windy ( )
```

Definition at line 21 of file windy.cpp.

```
21     {
22     int agentCount = 30;
23     float maxForce = 0.3;
24     float maxSpeed = 0.6;
25
26     name = "flow field";
27     createAgent(RANDOM, &agentCount, &maxForce, &maxSpeed);
28     callback = reinterpret_cast <void(*)()> ( (void *)(&loop) );
29 }
```

5.21.3 Member Function Documentation

5.21.3.1 loop()

```
void windy::loop ( ) [static]
```

Definition at line 10 of file windy.cpp.

```
10 {  
11     for(auto it = agents.begin(); it < agents.end(); it++){  
12         flow = flowField(pvector(GRAVITY));  
13         (*it).force = behavior.inFlowField(*it, flow);  
14  
15         flow = flowField(pvector(WIND_WEST));  
16         (*it).force += behavior.inFlowField(*it, flow);  
17     }  
18     refresh();  
19 }
```

5.21.4 Member Data Documentation

5.21.4.1 flow

```
flowField windy::flow [static]
```

Definition at line 13 of file windy.h.

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

- include/[windy.h](#)
- src/[windy.cpp](#)

Chapter 6

File Documentation

6.1 include/agent.h File Reference

```
#include "point.h"
#include "color.h"
#include "flowField.h"
#include <vector>
#include <string>
Include dependency graph for agent.h:
```

6.2 include/color.h File Reference

color class used for agent, path, wall etc. color

```
#include <vector>
Include dependency graph for color.h: This graph shows which files directly or indirectly include this file:
```

Classes

- class [color](#)

Enumerations

- enum [num](#) {
 [BLACK](#) =0, [BLUE](#), [GREEN](#), [CYAN](#),
 [RED](#), [MAGENDA](#), [YELLOW](#), [WHITE](#) }
 used to get color from colors vector

6.2.1 Detailed Description

color class used for agent, path, wall etc. color

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

13.05.2021

6.2.2 Enumeration Type Documentation

6.2.2.1 num

enum [num](#)

used to get color from colors vector

color names for fundamental colors

Enumerator

BLACK	
BLUE	
GREEN	
CYAN	
RED	
MAGENDA	
YELLOW	
WHITE	

Definition at line 18 of file color.h.

```
18 { BLACK=0, BLUE, GREEN, CYAN, RED, MAGENDA, YELLOW, WHITE };
```

6.3 include/evade.h File Reference

```
#include "scenario.h"
```

```
#include <vector>
```

Include dependency graph for evade.h: This graph shows which files directly or indirectly include this file:

Classes

- class [evade](#)

6.4 include/flee.h File Reference

```
#include "scenario.h"
```

```
#include <vector>
```

Include dependency graph for flee.h: This graph shows which files directly or indirectly include this file:

Classes

- class [flee](#)

6.5 include/flock.h File Reference

```
#include "scenario.h"
#include <vector>
```

Include dependency graph for flock.h: This graph shows which files directly or indirectly include this file:

Classes

- class [flock](#)

6.6 include/flowField.h File Reference

[flowField](#) class, screen can be filled with a force for each pixel

```
#include "pvector.h"
```

Include dependency graph for flowField.h: This graph shows which files directly or indirectly include this file:

Classes

- class [flowField](#)

Macros

- #define [WIDTH](#) 34
- #define [HEIGHT](#) 34
- #define [WIND_WEST](#) 0.1, 0.0
- #define [GRAVITY](#) 0.0, -0.1

6.6.1 Detailed Description

[flowField](#) class, screen can be filled with a force for each pixel

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

13.05.2021

6.6.2 Macro Definition Documentation

6.6.2.1 GRAVITY

```
#define GRAVITY 0.0, -0.1
```

Definition at line 16 of file flowField.h.

6.6.2.2 HEIGHT

```
#define HEIGHT 34
```

Definition at line 13 of file flowField.h.

6.6.2.3 WIDTH

```
#define WIDTH 34
```

Definition at line 12 of file flowField.h.

6.6.2.4 WIND_WEST

```
#define WIND_WEST 0.1, 0.0
```

Definition at line 15 of file flowField.h.

6.7 include/graphics.h File Reference

```
#include "agent.h"
```

```
#include "path.h"
```

Include dependency graph for graphics.h: This graph shows which files directly or indirectly include this file:

Classes

- class [graphics](#)

Macros

- #define [WIDTH](#) 34
- #define [HEIGHT](#) 34
- #define [ESC](#) 27
- #define [PI](#) 3.14159265

6.7.1 Macro Definition Documentation

6.7.1.1 ESC

```
#define ESC 27
```

Definition at line 9 of file graphics.h.

6.7.1.2 HEIGHT

```
#define HEIGHT 34
```

Definition at line 7 of file graphics.h.

6.7.1.3 PI

```
#define PI 3.14159265
```

Definition at line 10 of file graphics.h.

6.7.1.4 WIDTH

```
#define WIDTH 34
```

Definition at line 6 of file graphics.h.

6.8 include/mouseFollower.h File Reference

```
#include "scenario.h"  
#include <vector>
```

Include dependency graph for mouseFollower.h: This graph shows which files directly or indirectly include this file:

Classes

- class [mouseFollower](#)

6.9 include/obstacle.h File Reference

circular obstacles for agent avoidance behaviors

```
#include "point.h"
```

Include dependency graph for obstacle.h: This graph shows which files directly or indirectly include this file:

Classes

- class [obstacle](#)

6.9.1 Detailed Description

circular obstacles for agent avoidance behaviors

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.10 include/obstacleAvoidance.h File Reference

```
#include "scenario.h"  
#include "obstacle.h"  
#include <vector>
```

Include dependency graph for obstacleAvoidance.h: This graph shows which files directly or indirectly include this file:

Classes

- class [obstacleAvoidance](#)

6.11 include/path.h File Reference

path class used for path following steering behaviors.

```
#include "point.h"  
#include <vector>
```

Include dependency graph for path.h: This graph shows which files directly or indirectly include this file:

Classes

- class [path](#)

6.11.1 Detailed Description

path class used for path following steering behaviors.

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.12 include/pathFollower.h File Reference

```
#include "scenario.h"
#include <vector>
```

Include dependency graph for pathFollower.h: This graph shows which files directly or indirectly include this file:

Classes

- class [pathFollower](#)

6.13 include/point.h File Reference

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

Include dependency graph for point.h: This graph shows which files directly or indirectly include this file:

Classes

- class [point](#)

6.14 include/prison.h File Reference

```
#include "scenario.h"
#include <vector>
```

Include dependency graph for prison.h: This graph shows which files directly or indirectly include this file:

Classes

- class [prison](#)

6.15 include/pursuit.h File Reference

```
#include "scenario.h"  
#include <vector>
```

Include dependency graph for pursuit.h: This graph shows which files directly or indirectly include this file:

Classes

- class [pursuit](#)

6.16 include/pvector.h File Reference

```
#include <string>
```

Include dependency graph for pvector.h: This graph shows which files directly or indirectly include this file:

Classes

- class [pvector](#)

Macros

- #define [PI](#) 3.14159265

6.16.1 Macro Definition Documentation

6.16.1.1 PI

```
#define PI 3.14159265
```

Definition at line 5 of file pvector.h.

6.17 include/random.h File Reference

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

Classes

- class [random](#)

6.18 include/scenario.h File Reference

```
#include "agent.h"
#include "graphics.h"
#include "steeringBehavior.h"
#include <vector>
```

Include dependency graph for scenario.h: This graph shows which files directly or indirectly include this file:

Classes

- class [scenario](#)

Enumerations

- enum [types](#) { [RANDOM](#) =0, [STATIC](#), [TROOP](#) }

6.18.1 Enumeration Type Documentation

6.18.1.1 types

```
enum types
```

Enumerator

RANDOM	
STATIC	
TROOP	

Definition at line 10 of file scenario.h.

```
10 { RANDOM=0, STATIC, TROOP };
```

6.19 include/steeringBehavior.h File Reference

```
#include "flowField.h"
#include <vector>
#include "graphics.h"
#include "obstacle.h"
```

Include dependency graph for steeringBehavior.h: This graph shows which files directly or indirectly include this file:

Classes

- class [steeringBehavior](#)

Macros

- `#define CIRCLE_DISTANCE 0.1`
- `#define CIRCLE_RADIUS 0.4`
- `#define FOLLOW_MOUSE 1`
- `#define STAY_IN_FIELD 2`
- `#define IN_FLOW_FIELD 3`
- `#define AVOID_OBSTACLE 4`
- `#define STAY_IN_PATH 5`
- `#define FLOCK 6`
- `#define WANDER 7`
- `#define FLEE 8`
- `#define PURSUIT 9`
- `#define EVADE 10`

6.19.1 Macro Definition Documentation

6.19.1.1 AVOID_OBSTACLE

```
#define AVOID_OBSTACLE 4
```

Definition at line 14 of file steeringBehavior.h.

6.19.1.2 CIRCLE_DISTANCE

```
#define CIRCLE_DISTANCE 0.1
```

Definition at line 8 of file steeringBehavior.h.

6.19.1.3 CIRCLE_RADIUS

```
#define CIRCLE_RADIUS 0.4
```

Definition at line 9 of file steeringBehavior.h.

6.19.1.4 EVADE

```
#define EVADE 10
```

Definition at line 20 of file steeringBehavior.h.

6.19.1.5 FLEE

```
#define FLEE 8
```

Definition at line 18 of file steeringBehavior.h.

6.19.1.6 FLOCK

```
#define FLOCK 6
```

Definition at line 16 of file steeringBehavior.h.

6.19.1.7 FOLLOW_MOUSE

```
#define FOLLOW_MOUSE 1
```

Definition at line 11 of file steeringBehavior.h.

6.19.1.8 IN_FLOW_FIELD

```
#define IN_FLOW_FIELD 3
```

Definition at line 13 of file steeringBehavior.h.

6.19.1.9 PURSUIT

```
#define PURSUIT 9
```

Definition at line 19 of file steeringBehavior.h.

6.19.1.10 STAY_IN_FIELD

```
#define STAY_IN_FIELD 2
```

Definition at line 12 of file steeringBehavior.h.

6.19.1.11 STAY_IN_PATH

```
#define STAY_IN_PATH 5
```

Definition at line 15 of file steeringBehavior.h.

6.19.1.12 WANDER

```
#define WANDER 7
```

Definition at line 17 of file steeringBehavior.h.

6.20 include/wander.h File Reference

```
#include "scenario.h"
```

```
#include <vector>
```

Include dependency graph for wander.h: This graph shows which files directly or indirectly include this file:

Classes

- class [wander](#)

6.21 include/windy.h File Reference

```
#include "scenario.h"
```

```
#include "flowField.h"
```

```
#include <vector>
```

Include dependency graph for windy.h: This graph shows which files directly or indirectly include this file:

Classes

- class [windy](#)

6.22 main.cpp File Reference

```
#include <iostream>
```

```
#include "mouseFollower.h"
```

```
#include "prison.h"
```

```
#include "windy.h"
```

```
#include "wander.h"
```

```
#include "pursuit.h"
```

```
#include "flee.h"
```

```
#include "scenario.h"
```

```
#include "evade.h"
```

```
#include "flock.h"
```

```
#include "pathFollower.h"
```

```
#include "obstacleAvoidance.h"
```

Include dependency graph for main.cpp:

Functions

- void `menu()`
- int `main` (int argc, char **argv)

Variables

- int `mode`

6.22.1 Function Documentation

6.22.1.1 `main()`

```
int main (  
    int argc,  
    char ** argv )
```

Definition at line 32 of file main.cpp.

```
32     {  
33     menu();  
34  
35     scenario* sc;  
36  
37     if(mode == FOLLOW_MOUSE) {  
38         *sc = mouseFollower();  
39     }  
40     else if(mode == STAY_IN_FIELD) {  
41         *sc = prison();  
42     }  
43     else if(mode == IN_FLOW_FIELD) {  
44         *sc = windy();  
45     }  
46     else if(mode == WANDER) {  
47         *sc = wander();  
48     }  
49     else if(mode == PURSUIT) {  
50         *sc = pursuit();  
51     }  
52     else if(mode == FLEE) {  
53         *sc = flee();  
54     }  
55     else if(mode == EVADE) {  
56         *sc = evade();  
57     }  
58     else if(mode == FLOCK) {  
59         *sc = flock();  
60     }  
61     else if(mode == STAY_IN_PATH) {  
62         *sc = pathFollower();  
63     }  
64     else if(mode == AVOID_OBSTACLE) {  
65         *sc = obstacleAvoidance();  
66     }  
67  
68     sc->initGL(&argc, argv);  
69  
70     return 0;  
71 }
```

Here is the call graph for this function:

6.22.1.2 menu()

```
void menu ( )
```

Definition at line 18 of file main.cpp.

```
18     {
19         cout << "Follow Mouse      : 1" << endl;
20         cout << "Stay in Field      : 2" << endl;
21         cout << "In Flow Field    : 3" << endl;
22         cout << "OBSTACLE AVOIDANCE : 4" << endl;
23         cout << "Stay in Path      : 5" << endl;
24         cout << "FLOCK           : 6" << endl;
25         cout << "WANDER          : 7" << endl;
26         cout << "FLEE            : 8" << endl;
27         cout << "PURSUIT         : 9" << endl;
28         cout << "EVADE           : 10" << endl;
29         cin >> mode;
30     }
```

Here is the caller graph for this function:

6.22.2 Variable Documentation

6.22.2.1 mode

```
int mode
```

Definition at line 16 of file main.cpp.

6.23 README.md File Reference

6.24 src/agent.cpp File Reference

```
#include "agent.h"
#include "pvector.h"
#include "graphics.h"
#include "random.h"
#include <iostream>
Include dependency graph for agent.cpp:
```

6.25 src/color.cpp File Reference

color class implementation

```
#include "color.h"
#include <vector>
Include dependency graph for color.cpp:
```

6.25.1 Detailed Description

color class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

13.05.2021

6.26 src/evade.cpp File Reference

```
#include "scenario.h"
#include "evade.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for evade.cpp:
```

6.27 src/flee.cpp File Reference

```
#include "scenario.h"
#include "flee.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for flee.cpp:
```

6.28 src/flock.cpp File Reference

```
#include "scenario.h"
#include "flock.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for flock.cpp:
```

6.29 src/flowField.cpp File Reference

[flowField](#) class implementation

```
#include "flowField.h"
Include dependency graph for flowField.cpp:
```

6.29.1 Detailed Description

`flowField` class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

13.05.2021

6.30 src/graphics.cpp File Reference

```
#include "graphics.h"
#include <GL/glut.h>
#include <iostream>
#include "math.h"
Include dependency graph for graphics.cpp:
```

6.31 src/mouseFollower.cpp File Reference

```
#include "scenario.h"
#include "mouseFollower.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for mouseFollower.cpp:
```

6.32 src/obstacle.cpp File Reference

obstacle class implementation

```
#include "obstacle.h"
#include "graphics.h"
#include "point.h"
#include <vector>
Include dependency graph for obstacle.cpp:
```

6.32.1 Detailed Description

obstacle class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.33 src/obstacleAvoidance.cpp File Reference

```
#include "scenario.h"
#include "obstacleAvoidance.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for obstacleAvoidance.cpp:
```

6.34 src/path.cpp File Reference

path class implementation

```
#include "path.h"
#include "graphics.h"
Include dependency graph for path.cpp:
```

6.34.1 Detailed Description

path class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.35 src/pathFollower.cpp File Reference

```
#include "scenario.h"
#include "pathFollower.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for pathFollower.cpp:
```

6.36 src/point.cpp File Reference

```
#include "point.h"
#include "pvector.h"
#include <string>
#include <iostream>
Include dependency graph for point.cpp:
```

6.37 src/prison.cpp File Reference

```
#include "scenario.h"
#include "prison.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for prison.cpp:
```

6.38 src/pursuit.cpp File Reference

```
#include "scenario.h"
#include "pursuit.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for pursuit.cpp:
```

6.39 src/pvector.cpp File Reference

```
#include "pvector.h"
#include "math.h"
#include "point.h"
#include <iostream>
#include <string>
Include dependency graph for pvector.cpp:
```

6.40 src/random.cpp File Reference

```
#include "random.h"
#include <stdlib.h>
#include <iostream>
Include dependency graph for random.cpp:
```

6.41 src/scenario.cpp File Reference

```
#include "scenario.h"
#include "random.h"
#include <iostream>
Include dependency graph for scenario.cpp:
```

6.42 src/steeringBehavior.cpp File Reference

```
#include "steeringBehavior.h"
#include "pvector.h"
#include "agent.h"
#include "path.h"
#include "point.h"
#include <vector>
#include "graphics.h"
#include "math.h"
#include "obstacle.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for steeringBehavior.cpp:
```

6.43 src/wander.cpp File Reference

```
#include "scenario.h"
#include "wander.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for wander.cpp:
```

6.44 src/windy.cpp File Reference

```
#include "scenario.h"
#include "windy.h"
#include <iostream>
#include <GL/glut.h>
Include dependency graph for windy.cpp:
```

6.45 test/test_suites.cpp File Reference

```
#include <boost/test/included/unit_test.hpp>
#include "../include/pvector.h"
#include "../include/point.h"
#include <iostream>
Include dependency graph for test_suites.cpp:
```

Macros

- `#define BOOST_TEST_MODULE test_suites`

Functions

- [BOOST_AUTO_TEST_CASE](#) (s1t1)
- [BOOST_AUTO_TEST_CASE](#) (s1t2)
- [BOOST_AUTO_TEST_CASE](#) (s1t3)
- [BOOST_AUTO_TEST_CASE](#) (s1t4)
- [BOOST_AUTO_TEST_CASE](#) (s1t5)
- [BOOST_AUTO_TEST_CASE](#) (s1t6)
- [BOOST_AUTO_TEST_CASE](#) (s1t7)
- [BOOST_AUTO_TEST_CASE](#) (s1t8)
- [BOOST_AUTO_TEST_CASE](#) (s1t9)
- [BOOST_AUTO_TEST_CASE](#) (s2t1)
- [BOOST_AUTO_TEST_CASE](#) (s2t2)
- [BOOST_AUTO_TEST_CASE](#) (s2t3)

6.45.1 Macro Definition Documentation

6.45.1.1 BOOST_TEST_MODULE

```
#define BOOST_TEST_MODULE test_suites
```

Definition at line 1 of file test_suites.cpp.

6.45.2 Function Documentation

6.45.2.1 BOOST_AUTO_TEST_CASE() [1/12]

```
BOOST_AUTO_TEST_CASE (
    s1t1 )
```

Definition at line 11 of file test_suites.cpp.

```
11     {
12         pvector p1 = pvector(0, 4);
13         pvector p2 = pvector(3, 0);
14         pvector p3 = p1 + p2;
15         BOOST_CHECK(p3.magnitude() == 5);
16     }
```

Here is the call graph for this function:

6.45.2.2 BOOST_AUTO_TEST_CASE() [2/12]

```
BOOST_AUTO_TEST_CASE (
    s1t2 )
```

Definition at line 17 of file test_suites.cpp.

```
17     {
18         pvector p1 = pvector(1, 1);
19         p1.mul(3);
20         pvector p2 = pvector(3, 3);
21         BOOST_CHECK(p1 == p2);
22     }
```

Here is the call graph for this function:

6.45.2.3 BOOST_AUTO_TEST_CASE() [3/12]

```
BOOST_AUTO_TEST_CASE (
    slt3 )
```

Definition at line 23 of file test_suites.cpp.

```
23     {
24         pvector p1 = pvector(5, 5);
25         p1.div(5);
26         pvector p2 = pvector(1, 1);
27         BOOST_CHECK(p1 == p2);
28     }
```

Here is the call graph for this function:

6.45.2.4 BOOST_AUTO_TEST_CASE() [4/12]

```
BOOST_AUTO_TEST_CASE (
    slt4 )
```

Definition at line 29 of file test_suites.cpp.

```
29     {
30         pvector p1 = pvector(1, 4);
31         pvector p2 = pvector(3, 2);
32         float dotProduct = p1.dotProduct(p2);
33         BOOST_CHECK(dotProduct == 11);
34     }
```

Here is the call graph for this function:

6.45.2.5 BOOST_AUTO_TEST_CASE() [5/12]

```
BOOST_AUTO_TEST_CASE (
    slt5 )
```

Definition at line 35 of file test_suites.cpp.

```
35     {
36         pvector p1 = pvector(10, 10);
37         pvector p2 = pvector(0, 10);
38         float angle = p1.angleBetween(p2);
39         BOOST_CHECK(angle == 45);
40     }
```

Here is the call graph for this function:

6.45.2.6 BOOST_AUTO_TEST_CASE() [6/12]

```
BOOST_AUTO_TEST_CASE (
    slt6 )
```

Definition at line 41 of file test_suites.cpp.

```
41     {
42         pvector p1 = pvector(3, 4);
43         float angle = p1.getAngle();
44         BOOST_CHECK(angle < 53.2 && angle > 52.8);
45     }
```

Here is the call graph for this function:

6.45.2.7 BOOST_AUTO_TEST_CASE() [7/12]

```
BOOST_AUTO_TEST_CASE (
    slt7 )
```

Definition at line 46 of file test_suites.cpp.

```
46      {
47          pvector p1 = pvector(2, 2);
48          p1.normalize();
49          float range = 0.01;
50          BOOST_CHECK_CLOSE_FRACTION(0.707, p1.x, range);
51          BOOST_CHECK_CLOSE_FRACTION(0.707, p1.y, range);
52      }
```

Here is the call graph for this function:

6.45.2.8 BOOST_AUTO_TEST_CASE() [8/12]

```
BOOST_AUTO_TEST_CASE (
    slt8 )
```

Definition at line 53 of file test_suites.cpp.

```
53      {
54          pvector p1 = pvector(2, 2);
55          p1.limit(3);
56          float range = 0.01;
57          BOOST_CHECK_CLOSE_FRACTION(2.12, p1.x, range);
58          BOOST_CHECK_CLOSE_FRACTION(2.12, p1.y, range);
59      }
```

Here is the call graph for this function:

6.45.2.9 BOOST_AUTO_TEST_CASE() [9/12]

```
BOOST_AUTO_TEST_CASE (
    slt9 )
```

Definition at line 60 of file test_suites.cpp.

```
60      {
61          pvector p1 = pvector(1, 1);
62          p1 += pvector(1, 1);
63          BOOST_CHECK(p1 == pvector(2, 2));
64          p1 = pvector(1, 1) + pvector(3, 3);
65          BOOST_CHECK(p1 == pvector(4, 4));
66          p1 = pvector(4, 1) - pvector(3, 3);
67          BOOST_CHECK(p1 == pvector(1, -2));
68          p1 = pvector(4, 1) - point(3, 3);
69          BOOST_CHECK(p1 == pvector(1, -2));
70          p1 = pvector(4, 1) + point(3, 3);
71          BOOST_CHECK(p1 == pvector(7, 4));
72      }
```

Here is the call graph for this function:

6.45.2.10 BOOST_AUTO_TEST_CASE() [10/12]

```
BOOST_AUTO_TEST_CASE (
    s2t1 )
```

Definition at line 76 of file test_suites.cpp.

```
76      {
77          point p1 = point(1, 1);
78          p1.mul(3);
79          point p2 = point(3, 3);
80          BOOST_CHECK(p1 == p2);
81      }
```

Here is the call graph for this function:

6.45.2.11 BOOST_AUTO_TEST_CASE() [11/12]

```
BOOST_AUTO_TEST_CASE (
    s2t2 )
```

Definition at line 82 of file test_suites.cpp.

```
82     {
83         point p1 = point (4, 4);
84         p1.div(4);
85         point p2 = point (1, 1);
86         BOOST_CHECK(p1 == p2);
87     }
```

Here is the call graph for this function:

6.45.2.12 BOOST_AUTO_TEST_CASE() [12/12]

```
BOOST_AUTO_TEST_CASE (
    s2t3 )
```

Definition at line 88 of file test_suites.cpp.

```
88     {
89         point p1 = point (1,1) + point (3,3);
90         BOOST_CHECK(p1 == point (4,4));
91         p1 = point (1,1) + pvector (3,3);
92         BOOST_CHECK(p1 == point (4,4));
93         pvector p2 = point (1,1) - point (3,3);
94         BOOST_CHECK(p2 == pvector (-2,-2));
95     }
```

Here is the call graph for this function:

Index

- ~agent
 - agent, 10
- acceleration
 - agent, 11
- add
 - pvector, 46
- addPoint
 - path, 35
- agent, 9
 - ~agent, 10
 - acceleration, 11
 - agent, 10
 - arrive, 11
 - desiredVelocity, 11
 - fillColor, 11
 - force, 12
 - id, 12
 - mass, 12
 - maxForce, 12
 - maxSpeed, 12
 - name, 12
 - position, 13
 - r, 13
 - setFeatures, 10
 - steering, 13
 - targetPoint, 13
 - updatePosition, 10
 - velocity, 13
- agents
 - scenario, 53
- align
 - steeringBehavior, 55
- angleBetween
 - pvector, 46
- arrive
 - agent, 11
- avoid
 - steeringBehavior, 55
- AVOID_OBSTACLE
 - steeringBehavior.h, 74
- B
 - color, 16
- behavior
 - scenario, 53
- BLACK
 - color.h, 66
- BLUE
 - color.h, 66
- BOOST_AUTO_TEST_CASE
 - test_suites.cpp, 84–87
- BOOST_TEST_MODULE
 - test_suites.cpp, 84
- callback
 - scenario, 53
- CIRCLE_DISTANCE
 - steeringBehavior.h, 74
- CIRCLE_RADIUS
 - steeringBehavior.h, 74
- cohesion
 - steeringBehavior, 55
- color, 14
 - B, 16
 - color, 14
 - colors, 16
 - createColors, 15
 - G, 16
 - getColor, 15
 - R, 16
- color.h
 - BLACK, 66
 - BLUE, 66
 - CYAN, 66
 - GREEN, 66
 - MAGENDA, 66
 - num, 66
 - RED, 66
 - WHITE, 66
 - YELLOW, 66
- colors
 - color, 16
- createAgent
 - scenario, 52
- createColors
 - color, 15
- createObstacle
 - obstacleAvoidance, 33
- createPath
 - pathFollower, 37
- createRandomArray
 - random, 50
- CYAN
 - color.h, 66
- desiredVelocity
 - agent, 11
- div
 - point, 40

- pvector, 46
- dotProduct
 - pvector, 46
- drawAgent
 - graphics, 23
- drawCircle
 - graphics, 23
- drawLine
 - graphics, 24
- drawPath
 - graphics, 24
- drawPoint
 - graphics, 24
- drawText
 - graphics, 25
- drawWall
 - graphics, 25
- ESC
 - graphics.h, 69
- EVADE
 - steeringBehavior.h, 74
- evade, 17
 - evade, 17
 - loop, 18
 - steeringBehavior, 56
- fillColor
 - agent, 11
- FLEE
 - steeringBehavior.h, 74
- flee, 18
 - flee, 19
 - loop, 19
 - steeringBehavior, 56
- FLOCK
 - steeringBehavior.h, 75
- flock, 19
 - flock, 20
 - loop, 20
- flow
 - windy, 63
- flowField, 21
 - flowField, 21
 - getField, 22
- flowField.h
 - GRAVITY, 67
 - HEIGHT, 68
 - WIDTH, 68
 - WIND_WEST, 68
- FOLLOW_MOUSE
 - steeringBehavior.h, 75
- force
 - agent, 12
- forceInScreen
 - graphics, 25
- G
 - color, 16
- getAngle
 - pvector, 46
- getColor
 - color, 15
- getField
 - flowField, 22
- getMousePosition
 - graphics, 26
- getNormalPoint
 - point, 40
- graphics, 22
 - drawAgent, 23
 - drawCircle, 23
 - drawLine, 24
 - drawPath, 24
 - drawPoint, 24
 - drawText, 25
 - drawWall, 25
 - forceInScreen, 25
 - getMousePosition, 26
 - handleKeypress, 26
 - handleResize, 26
 - initGraphics, 26
 - mouseButton, 27
 - mouseMove, 27
 - refreshScene, 27
 - target_x, 28
 - target_y, 28
 - timerEvent, 28
- graphics.h
 - ESC, 69
 - HEIGHT, 69
 - PI, 69
 - WIDTH, 69
- GRAVITY
 - flowField.h, 67
- GREEN
 - color.h, 66
- handleKeypress
 - graphics, 26
- handleResize
 - graphics, 26
- HEIGHT
 - flowField.h, 68
 - graphics.h, 69
- id
 - agent, 12
- IN_FLOW_FIELD
 - steeringBehavior.h, 75
- include/agent.h, 65
- include/color.h, 65
- include/evade.h, 66
- include/flee.h, 66
- include/flock.h, 67
- include/flowField.h, 67
- include/graphics.h, 68
- include/mouseFollower.h, 69

- include/obstacle.h, 70
- include/obstacleAvoidance.h, 70
- include/path.h, 70
- include/pathFollower.h, 71
- include/point.h, 71
- include/prison.h, 71
- include/pursuit.h, 72
- include/pvector.h, 72
- include/random.h, 72
- include/scenario.h, 73
- include/steeringBehavior.h, 73
- include/wander.h, 76
- include/windy.h, 76
- inFlowField
 - steeringBehavior, 57
- initGL
 - scenario, 52
- initGraphics
 - graphics, 26
- limit
 - pvector, 47
- loop
 - evade, 18
 - flee, 19
 - flock, 20
 - mouseFollower, 29
 - obstacleAvoidance, 33
 - pathFollower, 38
 - prison, 43
 - pursuit, 44
 - wander, 61
 - windy, 62
- MAGENDA
 - color.h, 66
- magnitude
 - pvector, 47
- main
 - main.cpp, 77
- main.cpp, 76
 - main, 77
 - menu, 77
 - mode, 78
- mass
 - agent, 12
- maxForce
 - agent, 12
- maxSpeed
 - agent, 12
- menu
 - main.cpp, 77
- mode
 - main.cpp, 78
- mouseButton
 - graphics, 27
- mouseFollower, 29
 - loop, 29
 - mouseFollower, 29
- mouseMove
 - graphics, 27
- mul
 - point, 40
 - pvector, 47
- myColor
 - scenario, 53
- myPath
 - pathFollower, 38
- name
 - agent, 12
 - scenario, 53
- normalize
 - pvector, 47
- num
 - color.h, 66
- obstacle, 30
 - obstacle, 30, 31
 - p, 31
 - r, 31
- obstacleAvoidance, 32
 - createObstacle, 33
 - loop, 33
 - obstacleAvoidance, 32
 - obstacles, 33
- obstacles
 - obstacleAvoidance, 33
- operator+
 - point, 40
 - pvector, 48
- operator+=
 - pvector, 48
- operator-
 - point, 41
 - pvector, 48, 49
- operator==
 - point, 41
 - pvector, 49
- p
 - obstacle, 31
- path, 34
 - addPoint, 35
 - path, 34, 35
 - points, 36
 - width, 36
- pathFollower, 36
 - createPath, 37
 - loop, 38
 - myPath, 38
 - pathFollower, 37
- PI
 - graphics.h, 69
 - pvector.h, 72
- point, 38
 - div, 40
 - getNormalPoint, 40

- mul, 40
- operator+, 40
- operator-, 41
- operator==, 41
- point, 39
- print, 41
- x, 41
- y, 42
- points
 - path, 36
- position
 - agent, 13
- print
 - point, 41
 - pvector, 49
- prison, 42
 - loop, 43
 - prison, 42
- PURSUIT
 - steeringBehavior.h, 75
- pursuit, 43
 - loop, 44
 - pursuit, 44
 - steeringBehavior, 57
- pvector, 44
 - add, 46
 - angleBetween, 46
 - div, 46
 - dotProduct, 46
 - getAngle, 46
 - limit, 47
 - magnitude, 47
 - mul, 47
 - normalize, 47
 - operator+, 48
 - operator+==, 48
 - operator-, 48, 49
 - operator==, 49
 - print, 49
 - pvector, 45
 - x, 49
 - y, 50
- pvector.h
 - PI, 72
- R
 - color, 16
- r
 - agent, 13
 - obstacle, 31
- RANDOM
 - scenario.h, 73
- random, 50
 - createRandomArray, 50
- README.md, 78
- RED
 - color.h, 66
- refresh
 - scenario, 52
- refreshScene
 - graphics, 27
- scenario, 51
 - agents, 53
 - behavior, 53
 - callback, 53
 - createAgent, 52
 - initGL, 52
 - myColor, 53
 - name, 53
 - refresh, 52
 - scenario, 52
 - view, 54
- scenario.h
 - RANDOM, 73
 - STATIC, 73
 - TROOP, 73
 - types, 73
- seek
 - steeringBehavior, 57
- separation
 - steeringBehavior, 58
- setAngle
 - steeringBehavior, 58
- setFeatures
 - agent, 10
- src/agent.cpp, 78
- src/color.cpp, 78
- src/evade.cpp, 79
- src/flee.cpp, 79
- src/flock.cpp, 79
- src/flowField.cpp, 79
- src/graphics.cpp, 80
- src/mouseFollower.cpp, 80
- src/obstacle.cpp, 80
- src/obstacleAvoidance.cpp, 81
- src/path.cpp, 81
- src/pathFollower.cpp, 81
- src/point.cpp, 81
- src/prison.cpp, 82
- src/pursuit.cpp, 82
- src/pvector.cpp, 82
- src/random.cpp, 82
- src/scenario.cpp, 82
- src/steeringBehavior.cpp, 83
- src/wander.cpp, 83
- src/windy.cpp, 83
- STATIC
 - scenario.h, 73
- STAY_IN_FIELD
 - steeringBehavior.h, 75
- STAY_IN_PATH
 - steeringBehavior.h, 75
- stayInArea
 - steeringBehavior, 58
- stayInPath
 - steeringBehavior, 59
- stayInPath_2

- steeringBehavior, 59
- steering
 - agent, 13
- steeringBehavior, 54
 - align, 55
 - avoid, 55
 - cohesion, 55
 - evade, 56
 - flee, 56
 - inFlowField, 57
 - pursuit, 57
 - seek, 57
 - separation, 58
 - setAngle, 58
 - stayInArea, 58
 - stayInPath, 59
 - stayInPath_2, 59
 - wander, 60
- steeringBehavior.h
 - AVOID_OBSTACLE, 74
 - CIRCLE_DISTANCE, 74
 - CIRCLE_RADIUS, 74
 - EVADE, 74
 - FLEE, 74
 - FLOCK, 75
 - FOLLOW_MOUSE, 75
 - IN_FLOW_FIELD, 75
 - PURSUIT, 75
 - STAY_IN_FIELD, 75
 - STAY_IN_PATH, 75
 - WANDER, 76
- target_x
 - graphics, 28
- target_y
 - graphics, 28
- targetPoint
 - agent, 13
- test/test_suites.cpp, 83
- test_suites.cpp
 - BOOST_AUTO_TEST_CASE, 84–87
 - BOOST_TEST_MODULE, 84
- timerEvent
 - graphics, 28
- TROOP
 - scenario.h, 73
- types
 - scenario.h, 73
- updatePosition
 - agent, 10
- velocity
 - agent, 13
- view
 - scenario, 54
- WANDER
 - steeringBehavior.h, 76
- wander, 60
 - loop, 61
 - steeringBehavior, 60
 - wander, 61
- WHITE
 - color.h, 66
- WIDTH
 - flowField.h, 68
 - graphics.h, 69
- width
 - path, 36
- WIND_WEST
 - flowField.h, 68
- windy, 62
 - flow, 63
 - loop, 62
 - windy, 62
- x
 - point, 41
 - pvector, 49
- y
 - point, 42
 - pvector, 50
- YELLOW
 - color.h, 66