Autonomous Steering Agents

Generated by Doxygen 1.8.17

1 Intent 1

1.1 Dependencies . . . 1

1.2 Resources . . . 1

2 Todo List 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] . . . 14

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 . . . 15
5.2.4.1 B . . . 15
5.2.4.2 colors . . . 15
5.2.4.3 G . . . 16
5.2.4.4 R . . . 16
5.3 flowField Class Reference . . . 16
5.3.1 Detailed Description . . . 16
5.3.2 Constructor & Destructor Documentation . . . 16
5.3.2.1 flowField() [1/2] . . . 17
5.3.2.2 flowField() [2/2] . . . 17
5.3.3 Member Function Documentation . . . 17
5.3.3.1 getField() . . . 17
5.4 graphics Class Reference . . . 18
5.4.1 Detailed Description . . . 19
5.4.2 Member Function Documentation . . . 19
5.4.2.1 drawAgent() . . . 19
5.4.2.2 drawCircle() . . . 19
5.4.2.3 drawLine() . . . 20
5.4.2.4 drawPath() . . . 20
5.4.2.5 drawPoint() . . . 20
5.4.2.6 drawText() . . . 21
5.4.2.7 drawWall() . . . 21
5.4.2.8 forceInScreen() . . . 21
5.4.2.9 getMousePosition() . . . 22
5.4.2.10 handleKeypress() . . . 22
5.4.2.11 handleResize() . . . 22
5.4.2.12 initGraphics() . . . 22
5.4.2.13 mouseButton() . . . 23
5.4.2.14 mouseMove() . . . 23
5.4.2.15 refreshScene() . . . 23
5.4.2.16 timerEvent() . . . 23
5.4.3 Member Data Documentation . . . 24
5.4.3.1 target_x . . . 24
5.4.3.2 target_y . . . 24
5.5 obstacle Class Reference . . . 24
5.5.1 Detailed Description . . . 24
5.5.2 Constructor & Destructor Documentation . . . 25
5.5.2.1 obstacle() [1/2] . . . 25
5.5.2.2 obstacle() [2/2] . . . 25
5.5.3 Member Data Documentation . . . 25

5.5.3.1 p . . . 26
5.5.3.2 r . . . 26
5.6 path Class Reference . . . 26
5.6.1 Detailed Description . . . 27
5.6.2 Constructor & Destructor Documentation . . . 27
5.6.2.1 path() [1/2] . . . 27
5.6.2.2 path() [2/2] . . . 27
5.6.3 Member Function Documentation . . . 28
5.6.3.1 addPoint() . . . 28
5.6.3.2 createPath_1() . . . 28
5.6.3.3 createPath_2() . . . 28
5.6.4 Member Data Documentation . . . 29
5.6.4.1 points . . . 29
5.6.4.2 width . . . 29
5.7 point Class Reference . . . 29
5.7.1 Detailed Description . . . 30
5.7.2 Constructor & Destructor Documentation . . . 30
5.7.2.1 point() [1/2] . . . 30
5.7.2.2 point() [2/2] . . . 30
5.7.3 Member Function Documentation . . . 30
5.7.3.1 div() . . . 31
5.7.3.2 getNormalPoint() . . . 31
5.7.3.3 mul() . . . 31
5.7.3.4 operator+() [1/2] . . . 31
5.7.3.5 operator+() [2/2] . . . 32
5.7.3.6 operator-() . . . 32
5.7.3.7 operator==() . . . 32
5.7.3.8 print() . . . 32
5.7.4 Member Data Documentation . . . 32
5.7.4.1 x . . . 33
5.7.4.2 y . . . 33
5.8 pvector Class Reference . . . 33
5.8.1 Detailed Description . . . 34
5.8.2 Constructor & Destructor Documentation . . . 34
5.8.2.1 pvector() [1/2] . . . 34
5.8.2.2 pvector() [2/2] . . . 34
5.8.3 Member Function Documentation . . . 34
5.8.3.1 add() . . . 34
5.8.3.2 angleBetween() . . . 35
5.8.3.3 div() . . . 35
5.8.3.4 dotProduct() . . . 35
5.8.3.5 getAngle() . . . 35

5.8.3.6 limit() . . . 36
5.8.3.7 magnitude() . . . 36
5.8.3.8 mul() . . . 36
5.8.3.9 normalize() . . . 36
5.8.3.10 operator+() [1/2] . . . 37
5.8.3.11 operator+() [2/2] . . . 37
5.8.3.12 operator+=() . . . 37
5.8.3.13 operator-() [1/2] . . . 37
5.8.3.14 operator-() [2/2] . . . 38
5.8.3.15 operator==() . . . 38
5.8.3.16 print() . . . 38
5.8.4 Member Data Documentation . . . 38
5.8.4.1 x . . . 38
5.8.4.2 y . . . 39
5.9 random Class Reference . . . 39
5.9.1 Detailed Description . . . 39
5.9.2 Member Function Documentation . . . 39
5.9.2.1 createRandomArray() . . . 39
5.10 steeringBehavior Class Reference . . . 40
5.10.1 Detailed Description . . . 40
5.10.2 Member Function Documentation . . . 40
5.10.2.1 align() . . . 40
5.10.2.2 avoid() . . . 41
5.10.2.3 cohesion() . . . 41
5.10.2.4 evade() . . . 41
5.10.2.5 flee() . . . 42
5.10.2.6 inFlowField() . . . 42
5.10.2.7 pursuit() . . . 43
5.10.2.8 seek() . . . 43
5.10.2.9 separation() . . . 43
5.10.2.10 setAngle() . . . 44
5.10.2.11 stayInArea() . . . 44
5.10.2.12 stayInPath() . . . 44
5.10.2.13 stayInPath_2() . . . 45
5.10.2.14 wander() . . . 45

6 File Documentation **47**

6.1 /home/user/Desktop/mm/autonomousSteeringAgents/include/agent.h File Reference . . . 47
6.2 /home/user/Desktop/mm/autonomousSteeringAgents/include/color.h File Reference . . . 47
6.2.1 Enumeration Type Documentation . . . 47
6.2.1.1 num . . . 47
6.3 /home/user/Desktop/mm/autonomousSteeringAgents/include/flowField.h File Reference . . . 48

6.3.1 Detailed Description . . . 48
6.3.2 Macro Definition Documentation . . . 49
6.3.2.1 GRAVITY . . . 49
6.3.2.2 HEIGHT . . . 49
6.3.2.3 WIDTH . . . 49
6.3.2.4 WIND_WEST . . . 49
6.4 /home/user/Desktop/mm/autonomousSteeringAgents/include/graphics.h File Reference . . . 49
6.4.1 Macro Definition Documentation . . . 50
6.4.1.1 ESC . . . 50
6.4.1.2 HEIGHT . . . 50
6.4.1.3 PI . . . 50
6.4.1.4 WIDTH . . . 50
6.5 /home/user/Desktop/mm/autonomousSteeringAgents/include/obstacle.h File Reference . . . 50
6.5.1 Detailed Description . . . 51
6.6 /home/user/Desktop/mm/autonomousSteeringAgents/include/path.h File Reference . . . 51
6.6.1 Detailed Description . . . 51
6.7 /home/user/Desktop/mm/autonomousSteeringAgents/include/point.h File Reference . . . 51
6.8 /home/user/Desktop/mm/autonomousSteeringAgents/include/pvector.h File Reference . . . 52
6.8.1 Macro Definition Documentation . . . 52
6.8.1.1 PI . . . 52
6.9 /home/user/Desktop/mm/autonomousSteeringAgents/include/random.h File Reference . . . 52
6.10 /home/user/Desktop/mm/autonomousSteeringAgents/include/steeringBehavior.h File Reference . . . 53
6.10.1 Macro Definition Documentation . . . 53
6.10.1.1 AVOID_OBSTACLE . . . 53
6.10.1.2 CIRCLE_DISTANCE . . . 53
6.10.1.3 CIRCLE_RADIUS . . . 54
6.10.1.4 EVADE . . . 54
6.10.1.5 FLEE . . . 54
6.10.1.6 FLOCK . . . 54
6.10.1.7 FOLLOW_MOUSE . . . 54
6.10.1.8 IN_FLOW_FIELD . . . 54
6.10.1.9 PURSUIT . . . 55
6.10.1.10 STAY_IN_FIELD . . . 55
6.10.1.11 STAY_IN_PATH . . . 55
6.10.1.12 STAY_IN_PATH_2 . . . 55
6.10.1.13 WANDER . . . 55
6.11 /home/user/Desktop/mm/autonomousSteeringAgents/main.cpp File Reference . . . 55
6.11.1 Function Documentation . . . 56
6.11.1.1 createAgents() . . . 56
6.11.1.2 createObstacle() . . . 56
6.11.1.3 createRandomAgents() . . . 57
6.11.1.4 createTroop() . . . 57

6.11.1.5 init() . . . 57
6.11.1.6 loop() . . . 58
6.11.1.7 main() . . . 60
6.11.1.8 menu() . . . 60
6.11.2 Variable Documentation . . . 60
6.11.2.1 agents . . . 60
6.11.2.2 behavior . . . 60
6.11.2.3 flow . . . 61
6.11.2.4 mode . . . 61
6.11.2.5 myColor . . . 61
6.11.2.6 obstacles . . . 61
6.11.2.7 scenario . . . 61
6.11.2.8 view . . . 61
6.11.2.9 way . . . 62
6.12 /home/user/Desktop/mm/autonomousSteeringAgents/README.md File Reference . . . 62
6.13 /home/user/Desktop/mm/autonomousSteeringAgents/src/agent.cpp File Reference . . . 62
6.14 /home/user/Desktop/mm/autonomousSteeringAgents/src/color.cpp File Reference . . . 62
6.15 /home/user/Desktop/mm/autonomousSteeringAgents/src/flowField.cpp File Reference . . . 62
6.15.1 Detailed Description . . . 62
6.16 /home/user/Desktop/mm/autonomousSteeringAgents/src/graphics.cpp File Reference . . . 63
6.17 /home/user/Desktop/mm/autonomousSteeringAgents/src/obstacle.cpp File Reference . . . 63
6.17.1 Detailed Description . . . 63
6.18 /home/user/Desktop/mm/autonomousSteeringAgents/src/path.cpp File Reference . . . 63
6.18.1 Detailed Description . . . 64
6.19 /home/user/Desktop/mm/autonomousSteeringAgents/src/point.cpp File Reference . . . 64
6.20 /home/user/Desktop/mm/autonomousSteeringAgents/src/pvector.cpp File Reference . . . 64
6.21 /home/user/Desktop/mm/autonomousSteeringAgents/src/random.cpp File Reference . . . 64
6.22 /home/user/Desktop/mm/autonomousSteeringAgents/src/steeringBehavior.cpp File Reference . . . 65
6.23 /home/user/Desktop/mm/autonomousSteeringAgents/unit_test/test_suites.cpp File Reference . . . 65
6.23.1 Macro Definition Documentation . . . 65
6.23.1.1 BOOST_TEST_MODULE . . . 66
6.23.2 Function Documentation . . . 66
6.23.2.1 BOOST_AUTO_TEST_CASE() [1/12] . . . 66
6.23.2.2 BOOST_AUTO_TEST_CASE() [2/12] . . . 66
6.23.2.3 BOOST_AUTO_TEST_CASE() [3/12] . . . 66
6.23.2.4 BOOST_AUTO_TEST_CASE() [4/12] . . . 67
6.23.2.5 BOOST_AUTO_TEST_CASE() [5/12] . . . 67
6.23.2.6 BOOST_AUTO_TEST_CASE() [6/12] . . . 67
6.23.2.7 BOOST_AUTO_TEST_CASE() [7/12] . . . 67
6.23.2.8 BOOST_AUTO_TEST_CASE() [8/12] . . . 68
6.23.2.9 BOOST_AUTO_TEST_CASE() [9/12] . . . 68
6.23.2.10 BOOST_AUTO_TEST_CASE() [10/12] . . . 68

6.23.2.11 BOOST_AUTO_TEST_CASE() [11/12] 68
6.23.2.12 BOOST_AUTO_TEST_CASE() [12/12] 69

Index **71**

Chapter 1

Intent

1- implementing Craig Raynolds 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

Todo List

Member path::createPath_1 ()

move this routine to client side

Member path::createPath_2 ()

move this routine to client side

Chapter 3

Class Index

3.1 Class List

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

agent 9
color 14
flowField 16
graphics 18
obstacle 24
path 26
point 29
pvector 33
random 39
steeringBehavior 40

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/user/Desktop/mm/autonomousSteeringAgents/main.cpp . . . 55
/home/user/Desktop/mm/autonomousSteeringAgents/include/agent.h . . . 47
/home/user/Desktop/mm/autonomousSteeringAgents/include/color.h . . . 47
/home/user/Desktop/mm/autonomousSteeringAgents/include/flowField.h
FlowField class, screen can be filled with a force for each pixel . . . 48
/home/user/Desktop/mm/autonomousSteeringAgents/include/graphics.h . . . 49
/home/user/Desktop/mm/autonomousSteeringAgents/include/obstacle.h
Circular obstacles for agent avoidance behaviors . . . 50
/home/user/Desktop/mm/autonomousSteeringAgents/include/path.h
Path class used for path following steering behaviors . . . 51
/home/user/Desktop/mm/autonomousSteeringAgents/include/point.h . . . 51
/home/user/Desktop/mm/autonomousSteeringAgents/include/pvector.h . . . 52
/home/user/Desktop/mm/autonomousSteeringAgents/include/random.h . . . 52
/home/user/Desktop/mm/autonomousSteeringAgents/include/steeringBehavior.h . . . 53
/home/user/Desktop/mm/autonomousSteeringAgents/src/agent.cpp . . . 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/color.cpp . . . 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/flowField.cpp
FlowField class implementation . . . 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/graphics.cpp . . . 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/obstacle.cpp
Obstacle class implementation . . . 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/path.cpp
Path class implementation . . . 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/point.cpp . . . 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/pvector.cpp . . . 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/random.cpp . . . 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/steeringBehavior.cpp . . . 65
/home/user/Desktop/mm/autonomousSteeringAgents/unit_test/test_suites.cpp . . . 65

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 (int mode, 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 (
              int mode,
              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:

- /home/user/Desktop/mm/autonomousSteeringAgents/include/agent.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/agent.cpp

5.2 color Class Reference

`#include <color.h>`

Collaboration diagram for color:

Public Member Functions

- color (float r, float g, float b)
- color ()
- void createColors ()
- color getColor (int i)

Public Attributes

- float R
- float G
- float B
- vector< color > colors

5.2.1 Detailed Description

Definition at line 9 of file color.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 color() [1/2]

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

Definition at line 6 of file color.cpp.

```
6                                       {
7     R = r;
8     G = g;
9     B = b;
10 }
```

5.2.2.2 color() [2/2]

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

Definition at line 16 of file color.cpp.

```
16 {}
```

5.2.3 Member Function Documentation

5.2.3.1 createColors()

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

Definition at line 18 of file color.cpp.

```
18                          { //TODO: move to colors
19     colors.push_back(color(0.0, 0.0, 0.0));
20     colors.push_back(color(0.0, 0.0, 1.0));
21     colors.push_back(color(0.0, 1.0, 0.0));
22     colors.push_back(color(0.0, 1.0, 1.0));
23     colors.push_back(color(1.0, 0.0, 0.0));
24     colors.push_back(color(1.0, 0.0, 1.0));
25     colors.push_back(color(1.0, 1.0, 0.0));
26     colors.push_back(color(1.0, 1.0, 1.0));
27 }
```

Here is the caller graph for this function:

5.2.3.2 getColor()

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

Definition at line 12 of file color.cpp.

```
12                                {
13     return colors.at(i);
14 }
```

Here is the caller graph for this function:

5.2.4 Member Data Documentation

5.2.4.1 B

```
float color::B
```

Definition at line 17 of file color.h.

5.2.4.2 colors

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

Definition at line 18 of file color.h.

5.2.4.3 G

```
float color::G
```

Definition at line 16 of file color.h.

5.2.4.4 R

```
float color::R
```

Definition at line 15 of file color.h.

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/color.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/color.cpp

5.3 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

5.3.1 Detailed Description

Definition at line 18 of file flowField.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 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 {}
```

5.3.2.2 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    uniformVectorField(p);
13 }
```

5.3.3 Member Function Documentation

5.3.3.1 getField()

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

get force for individual pixel

get force for a specific position

Parameters

x	x cprovidesoordinate
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:

- /home/user/Desktop/mm/autonomousSteeringAgents/include/flowField.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/flowField.cpp

5.4 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

5.4.1 Detailed Description

Definition at line 15 of file graphics.h.

5.4.2 Member Function Documentation

5.4.2.1 drawAgent()

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

Definition at line 160 of file graphics.cpp.

```
160                                                 {
161     glPushMatrix();
162     glTranslatef(agent.position.x, agent.position.y, 0.0f);
163     glRotatef(agent.velocity.getAngle(), 0.0f, 0.0f, 1.0f);
164     glBegin(GL_TRIANGLES);
165     glColor3f( color.R, color.G, color.B);
166     glVertex3f( 1.0f,  0.0f, 0.0f);
167     glVertex3f(-1.0f,  0.5f, 0.0f);
168     glVertex3f(-1.0f, -0.5f, 0.0f);
169     glEnd();
170     glPopMatrix();
171 }
```

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

5.4.2.2 drawCircle()

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

Definition at line 122 of file graphics.cpp.

```
122                                             {
123    glBegin(GL_LINE_STRIP);
124    glLineWidth(2);
125    for (int i = 0; i <= 300; i++) {
126      float angle = 2 * PI * i / 300;
127      float x = cos(angle) * radius;
128      float y = sin(angle) * radius;
129      glVertex2d(p.x + x, p.y + y);
130    }
131    glEnd();
132 }
```

Here is the caller graph for this function:

5.4.2.3 drawLine()

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

Definition at line 113 of file graphics.cpp.

```
113                                                 {
114     glColor3f( cl.R, cl.G, cl.B);
115     glLineWidth(2);
116     glBegin(GL_LINES);
117     glVertex2f(p1.x, p1.y);
118     glVertex2f(p2.x, p2.y);
119     glEnd();
120 }
```

5.4.2.4 drawPath()

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

Definition at line 100 of file graphics.cpp.

```
100                                             {
101     point p1, p2;
102     for(auto it = path.points.begin(); it < path.points.end()-1; it++){
103         p1 = point((*it).x, (*it).y - path.width/2) ;
104         p2 = point((*(it+1)).x, (*(it+1)).y - path.width/2);
105         drawLine(p1, p2, color.getColor(BLUE));
106
107         p1 = point((*it).x, (*it).y + path.width/2) ;
108         p2 = point((*(it+1)).x, (*(it+1)).y + path.width/2);
109         drawLine(p1, p2, color.getColor(BLUE));
110     }
111 }
```

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

5.4.2.5 drawPoint()

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

Definition at line 134 of file graphics.cpp.

```
134                                 {
135     glColor3f(1,1,1);
136     glPointSize(4.0);
137     glBegin(GL_POINTS);
138     glVertex2f(p.x, p.y);
139     glEnd();
140 }
```

Here is the caller graph for this function:

5.4.2.6 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:

5.4.2.7 drawWall()

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

Definition at line 142 of file graphics.cpp.

```
142                                                    {
143     point p1 {-border,  border};
144     point p2 { border,  border};
145     drawLine(p1, p2, color.getColor(BLUE));
146
147     p1 = point ( border,  border);
148     p2 = point ( border, -border);
149     drawLine(p1, p2, color.getColor(BLUE));
150
151     p1 = point (  border, -border);
152     p2 = point ( -border, -border);
153     drawLine(p1, p2, color.getColor(BLUE));
154
155     p1 = point (-border,  border);
156     p2 = point (-border, -border);
157     drawLine(p1, p2, color.getColor(BLUE));
158 }
```

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

5.4.2.8 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 }
```

Here is the caller graph for this function:

5.4.2.9 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: Here is the caller graph for this function:

5.4.2.10 handleKeypress()

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

Definition at line 97 of file graphics.cpp.

```
97                                                          {
98     if (key == ESC){ exit(0); }
99 }
```

Here is the caller graph for this function:

5.4.2.11 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:

5.4.2.12 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: Here is the caller graph for this function:

5.4.2.13 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 }
```

Here is the caller graph for this function:

5.4.2.14 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:

5.4.2.15 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 }
```

Here is the caller graph for this function:

5.4.2.16 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:

5.4.3 Member Data Documentation

5.4.3.1 target_x

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

Definition at line 33 of file graphics.h.

5.4.3.2 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:

- /home/user/Desktop/mm/autonomousSteeringAgents/include/graphics.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/graphics.cpp

5.5 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

5.5.1 Detailed Description

Definition at line 12 of file obstacle.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 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 {}
```

5.5.2.2 obstacle() [2/2]

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

Constructor.

Create a new obstacle object.

Parameters

p	center of the circular obstacle
r	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 }
```

5.5.3 Member Data Documentation

5.5.3.1 p

```
point obstacle::p
```

x and y coordinates

center point of the obstacle

Definition at line 34 of file obstacle.h.

5.5.3.2 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:

- /home/user/Desktop/mm/autonomousSteeringAgents/include/obstacle.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/obstacle.cpp

5.6 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
- void createPath_1 ()
- void createPath_2 ()

Public Attributes

- vector< point > points
  points added to the path
- int width
  defines width of the path

5.6.1 Detailed Description

Definition at line 15 of file path.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 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 }
```

5.6.2.2 path() [2/2]

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

Constructor.

Create a new path object.

Parameters

width	The width of the path.

See also

path()

Definition at line 21 of file path.cpp.

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

5.6.3 Member Function Documentation

5.6.3.1 addPoint()

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

adds a new point to the path

Used when customizing path

Parameters

point	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:

5.6.3.2 createPath_1()

```
void path::createPath_1 ( )
```

Used when customizing path

Todo move this routine to client side

Definition at line 35 of file path.cpp.

```
36 {
37     width = 6;
38     point start = point(-WIDTH-5,  HEIGHT-40);
39     point end   = point( WIDTH+5, -HEIGHT+40);
40     this->addPoint(start);
41     this->addPoint(end);
42 }
```

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

5.6.3.3 createPath_2()

```
void path::createPath_2 ( )
```

Used when customizing path

Todo move this routine to client side

Definition at line 26 of file path.cpp.

```
27 {
28     width = 8;
29     this->addPoint(point(-40,  5));
30     this->addPoint(point(-14, 15));
31     this->addPoint(point( 10,  7));
32     this->addPoint(point( 40, 12));
33 }
```

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

5.6.4 Member Data Documentation

5.6.4.1 points

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

points added to the path

path is created from these points

Definition at line 55 of file path.h.

5.6.4.2 width

```
int path::width
```

defines width of the path

path width

Definition at line 61 of file path.h.

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/path.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/path.cpp

5.7 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)

Static Public Member Functions

- static point getNormalPoint (point predicted, point start, point end)

Public Attributes

- float x
- float y

5.7.1 Detailed Description

Definition at line 8 of file point.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 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 }
```

5.7.2.2 point() [2/2]

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

Definition at line 13 of file point.cpp.

```
13 {}
```

Here is the caller graph for this function:

5.7.3 Member Function Documentation

5.7.3.1 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:

5.7.3.2 getNormalPoint()

```
point point::getNormalPoint (
            point predicted,
            point start,
            point end )  [static]
```

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     return normalPoint;
61 }
```

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

5.7.3.3 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:

5.7.3.4 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 }
```

5.7.3.5 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 }
```

5.7.3.6 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 }
```

5.7.3.7 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 }
```

5.7.3.8 print()

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

Definition at line 63 of file point.cpp.

```
63                                  {
64     cout « " " « s « " " « x « " " « y « endl;
65 }
```

5.7.4 Member Data Documentation

5.7.4.1 x

```
float point::x
```

Definition at line 10 of file point.h.

5.7.4.2 y

```
float point::y
```

Definition at line 11 of file point.h.

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/point.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/point.cpp

5.8 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

5.8.1 Detailed Description

Definition at line 11 of file pvector.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 pvector() [1/2]

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

Definition at line 25 of file pvector.cpp.

```
25 {}
```

5.8.2.2 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 }
```

5.8.3 Member Function Documentation

5.8.3.1 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 }
```

5.8.3.2 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:

5.8.3.3 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:

5.8.3.4 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:

5.8.3.5 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:

5.8.3.6 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:

5.8.3.7 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:

5.8.3.8 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:

5.8.3.9 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:

5.8.3.10 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 }
```

5.8.3.11 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 }
```

5.8.3.12 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 }
```

5.8.3.13 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 }
```

5.8.3.14 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 }
```

5.8.3.15 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 }
```

5.8.3.16 print()

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

Definition at line 102 of file pvector.cpp.

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

5.8.4 Member Data Documentation

5.8.4.1 x

```
float pvector::x
```

Definition at line 13 of file pvector.h.

5.8.4.2 y

```
float pvector::y
```

Definition at line 14 of file pvector.h.

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/pvector.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/pvector.cpp

5.9 random Class Reference

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

Collaboration diagram for random:

Static Public Member Functions

- static void createRandomArray (int *arr, int size)

5.9.1 Detailed Description

Definition at line 3 of file random.h.

5.9.2 Member Function Documentation

5.9.2.1 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 }
```

Here is the caller graph for this function:

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/random.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/random.cpp

5.10 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)

5.10.1 Detailed Description

Definition at line 29 of file steeringBehavior.h.

5.10.2 Member Function Documentation

5.10.2.1 align()

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

Definition at line 105 of file steeringBehavior.cpp.

```
105                                                                    {
106     float neighborDist = 30; //TODO: magic numer
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: Here is the caller graph for this function:

5.10.2.2 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: Here is the caller graph for this function:

5.10.2.3 cohesion()

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

Definition at line 125 of file steeringBehavior.cpp.

```
125                                                                  {
126     float neighborDist = 20; //TODO: magic numer
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: Here is the caller graph for this function:

5.10.2.4 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: Here is the caller graph for this function:

5.10.2.5 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: Here is the caller graph for this function:

5.10.2.6 inFlowField()

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

Definition at line 235 of file steeringBehavior.cpp.

```
235                                                                               {
236     //pos_x, pos_y must be non negative integer
237     int pos_x = abs((int)agent.position.x) % WIDTH;
238     int pos_y = abs((int)agent.position.y) % HEIGHT;
239     //TODO: modification required for non uniform fields
240     return flow.getField(pos_x, pos_y);
241 }
```

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

5.10.2.7 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: Here is the caller graph for this function:

5.10.2.8 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 }
```

Here is the caller graph for this function:

5.10.2.9 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: Here is the caller graph for this function:

5.10.2.10 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 }
```

5.10.2.11 stayInArea()

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

Definition at line 243 of file steeringBehavior.cpp.

```
243                                                                 {
244     if(agent.position.x >= turnPoint){
245        agent.desiredVelocity = pvector( -agent.maxSpeed, agent.velocity.y );
246        agent.steering = agent.desiredVelocity - agent.velocity;
247        return agent.steering;
248     }
249     else if(agent.position.x <= -turnPoint){
250        agent.desiredVelocity = pvector( agent.maxSpeed, agent.velocity.y );
251        agent.steering = agent.desiredVelocity - agent.velocity;
252        return agent.steering;
253     }
254     else if(agent.position.y >= turnPoint){
255        agent.desiredVelocity = pvector( agent.velocity.x, -agent.maxSpeed );
256        agent.steering = agent.desiredVelocity - agent.velocity;
257        return agent.steering;
258     }
259     else if(agent.position.y <= -turnPoint){
260        agent.desiredVelocity = pvector( agent.velocity.x, agent.maxSpeed );
261        agent.steering = agent.desiredVelocity - agent.velocity;
262        return agent.steering;
263     }
264     return pvector(0,0);
265 }
```

Here is the caller graph for this function:

5.10.2.12 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 = point::getNormalPoint(predictedPos, start, end);
223     pvector b = end - start;
224     b.normalize();
225
226     pvector distance  = predictedPos - normalPoint;
227     agent.targetPoint = normalPoint  + b;
228     //view.drawLine(predictedPos, normalPoint);
229     //view.drawPoint(targetPoint);
230     if(distance.magnitude() > path.width / 8)
231       return seek(agent);
232     return pvector(0,0);
233 }
```

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

5.10.2.13 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 = point::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: Here is the caller graph for this function:

5.10.2.14 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: Here is the caller graph for this function:

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

- /home/user/Desktop/mm/autonomousSteeringAgents/include/steeringBehavior.h
- /home/user/Desktop/mm/autonomousSteeringAgents/src/steeringBehavior.cpp

Chapter 6

File Documentation

6.1 /home/user/Desktop/mm/autonomousSteering↩ Agents/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 /home/user/Desktop/mm/autonomousSteeringAgents/include/color.h File Reference

```
#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 }

6.2.1 Enumeration Type Documentation

6.2.1.1 num

```
enum num
```

Enumerator

BLACK	
BLUE	
GREEN	
CYAN	
RED	
MAGENDA	
YELLOW	
WHITE	

Definition at line 7 of file color.h.

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

6.3 /home/user/Desktop/mm/autonomousSteeringAgents/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

6.3.1 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

6.3.2 Macro Definition Documentation

6.3.2.1 GRAVITY

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

Definition at line 16 of file flowField.h.

6.3.2.2 HEIGHT

```
#define HEIGHT 34
```

Definition at line 13 of file flowField.h.

6.3.2.3 WIDTH

```
#define WIDTH 34
```

Definition at line 12 of file flowField.h.

6.3.2.4 WIND_WEST

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

Definition at line 15 of file flowField.h.

6.4 /home/user/Desktop/mm/autonomousSteering↩ Agents/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

6.4.1 Macro Definition Documentation

6.4.1.1 ESC

```
#define ESC 27
```

Definition at line 9 of file graphics.h.

6.4.1.2 HEIGHT

```
#define HEIGHT 34
```

Definition at line 7 of file graphics.h.

6.4.1.3 PI

```
#define PI 3.14159265
```

Definition at line 10 of file graphics.h.

6.4.1.4 WIDTH

```
#define WIDTH 34
```

Definition at line 6 of file graphics.h.

6.5 /home/user/Desktop/mm/autonomousSteering↩ Agents/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

6.5.1 Detailed Description

circular obstacles for agent avoidance behaviors

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.6 /home/user/Desktop/mm/autonomousSteeringAgents/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

6.6.1 Detailed Description

path class used for path following steering behaviors.

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.7 /home/user/Desktop/mm/autonomousSteeringAgents/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

6.8 /home/user/Desktop/mm/autonomousSteering↩ Agents/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

6.8.1 Macro Definition Documentation

6.8.1.1 PI

```
#define PI 3.14159265
```

Definition at line 5 of file pvector.h.

6.9 /home/user/Desktop/mm/autonomousSteering↩ Agents/include/random.h File Reference

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

Classes

- class random

6.10 /home/user/Desktop/mm/autonomousSteering↩ Agents/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 STAY_IN_PATH 4
- #define STAY_IN_PATH_2 5
- #define FLOCK 6
- #define WANDER 7
- #define FLEE 8
- #define PURSUIT 9
- #define EVADE 10
- #define AVOID_OBSTACLE 11

6.10.1 Macro Definition Documentation

6.10.1.1 AVOID_OBSTACLE

```
#define AVOID_OBSTACLE 11
```

Definition at line 21 of file steeringBehavior.h.

6.10.1.2 CIRCLE_DISTANCE

```
#define CIRCLE_DISTANCE 0.1
```

Definition at line 8 of file steeringBehavior.h.

6.10.1.3 CIRCLE_RADIUS

```
#define CIRCLE_RADIUS 0.4
```

Definition at line 9 of file steeringBehavior.h.

6.10.1.4 EVADE

```
#define EVADE 10
```

Definition at line 20 of file steeringBehavior.h.

6.10.1.5 FLEE

```
#define FLEE 8
```

Definition at line 18 of file steeringBehavior.h.

6.10.1.6 FLOCK

```
#define FLOCK 6
```

Definition at line 16 of file steeringBehavior.h.

6.10.1.7 FOLLOW_MOUSE

```
#define FOLLOW_MOUSE 1
```

Definition at line 11 of file steeringBehavior.h.

6.10.1.8 IN_FLOW_FIELD

```
#define IN_FLOW_FIELD 3
```

Definition at line 13 of file steeringBehavior.h.

6.10.1.9 PURSUIT

```
#define PURSUIT 9
```

Definition at line 19 of file steeringBehavior.h.

6.10.1.10 STAY_IN_FIELD

```
#define STAY_IN_FIELD 2
```

Definition at line 12 of file steeringBehavior.h.

6.10.1.11 STAY_IN_PATH

```
#define STAY_IN_PATH 4
```

Definition at line 14 of file steeringBehavior.h.

6.10.1.12 STAY_IN_PATH_2

```
#define STAY_IN_PATH_2 5
```

Definition at line 15 of file steeringBehavior.h.

6.10.1.13 WANDER

```
#define WANDER 7
```

Definition at line 17 of file steeringBehavior.h.

6.11 /home/user/Desktop/mm/autonomousSteeringAgents/main.cpp File Reference

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
#include "pvector.h"
#include "agent.h"
#include "point.h"
#include "color.h"
#include "graphics.h"
#include "flowField.h"
#include "obstacle.h"
#include "path.h"
#include "steeringBehavior.h"
#include <stdlib.h>
#include "random.h"
```

Include dependency graph for main.cpp:

Functions

- void menu ()
- void createRandomAgents (int agentCount, const float mForce, const float mSpeed)
- void createAgents ()
- void createTroop (int agentCount)
- void loop ()
- void createObstacle (vector< obstacle > &obstacles)
- void init (int *argv, char **argc, void(*callback)())
- int main (int argc, char **argv)

Variables

- int mode
- flowField flow
- graphics view
- path way
- steeringBehavior behavior
- string scenario
- vector< obstacle > obstacles
- color myColor
- vector< agent > agents

6.11.1 Function Documentation

6.11.1.1 createAgents()

```
void createAgents ( )
```

Definition at line 57 of file main.cpp.

```
57                  {
58     agent agent1 {-10.0,  0.0};
59     agent1.id = 1;
60     agent1.name = "gazelle";
61     agent1.fillColor = myColor.getColor(BLUE);
62     agent1.setFeatures(0.5, 0.2, 5, 1);
63     agents.push_back(agent1);
64
65     agent agent2 { 10.0,  0.0};
66     agent2.id = 2;
67     agent2.name = "lion";
68     agent2.fillColor = myColor.getColor(YELLOW);
69     agent2.setFeatures(0.4, 0.2, 5, 1);
70     agents.push_back(agent2);
71 }
```

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

6.11.1.2 createObstacle()

```
void createObstacle (
             vector< obstacle > & obstacles )
```

Definition at line 206 of file main.cpp.

```
206                                                {
207     obstacles.push_back(obstacle(point(0,0), 8));
208     obstacles.push_back(obstacle(point(-20,0), 3));
209     obstacles.push_back(obstacle(point(20,-10), 4));
210 }
```

Here is the caller graph for this function:

6.11.1.3 createRandomAgents()

```
void createRandomAgents (
              int agentCount,
              const float mForce,
              const float mSpeed )
```

Definition at line 43 of file main.cpp.

```
43                                                                              {
44     int size = MAX_NUMBER_OF_AGENTS * 2;
45     int arr[size];
46     random::createRandomArray(arr, size);
47     agent tempAgent {0, 0};
48     for(int i=0; i < agentCount * 2; i=i+2){
49        tempAgent.position.x = arr[i]   - WIDTH;
50        tempAgent.position.y = arr[i+1] - HEIGHT;
51        tempAgent.fillColor = myColor.colors.at( (i/2) % 8 );
52        tempAgent.setFeatures(mForce, mSpeed, 5, 1);
53        agents.push_back(tempAgent);
54     }
55 }
```

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

6.11.1.4 createTroop()

```
void createTroop (
              int agentCount )
```

Definition at line 73 of file main.cpp.

```
73                                {
74     //TODO: magic numbers
75     agent tempAgent {0, 0};
76     pvector location {-33, 33};
77
78     for(int i=0; i < agentCount; i++){
79         tempAgent.id = i;
80         tempAgent.velocity = pvector(0, 0);
81         tempAgent.position.x = location.x;
82         tempAgent.position.y = location.y;
83         tempAgent.targetPoint = tempAgent.position;
84
85         if( ((i+1) % 14) == 0){
86            location.y -= 5;
87            location.x  = -33;
88         }
89         else
90            location.x += 5;
91
92         tempAgent.fillColor = myColor.colors.at( (i/2) % 8 );
93         tempAgent.setFeatures(0.3, 0.3, 5, 1);
94         agents.push_back(tempAgent);
95     }
96 }
```

Here is the caller graph for this function:

6.11.1.5 init()

```
void init (
              int * argv,
              char ** argc,
              void(*)() callback )
```

Definition at line 212 of file main.cpp.

```
212                                                 {
213     srand(time(NULL));
214     myColor.createColors();
```

```
215
216     if(mode == STAY_IN_PATH){
217        way.createPath_1();
218        createRandomAgents(30, 0.6, 0.3);
219        scenario = "STAY IN PATH";
220     }
221     else if(mode == STAY_IN_PATH_2){
222        way.createPath_2();
223        createRandomAgents(40, 0.4, 0.2);
224        scenario = "STAY IN PATH 2";
225     }
226     else if(mode == FLEE){
227        createTroop(196);
228        scenario = "FLEE";
229     }
230     else if(mode == STAY_IN_FIELD){
231        createRandomAgents(30, 0.5, 0.5);
232        scenario = "STAY IN FIELD";
233     }
234     else if(mode == FOLLOW_MOUSE){
235        createRandomAgents(30, 0.6, 0.3);
236        scenario = "FOLLOW MOUSE";
237     }
238     else if(mode == FLOCK){
239        createRandomAgents(50, 1.0, 0.3);
240        scenario = "FLOCK";
241     }
242     else if(mode == WANDER){
243        createRandomAgents(30, 0.6, 0.3);
244        scenario = "WANDER";
245     }
246     else if(mode == IN_FLOW_FIELD){
247        createRandomAgents(30, 0.6, 0.3);
248        scenario = "IN FLOW FIELD";
249     }
250     else if(mode == PURSUIT){
251        createAgents();
252        scenario = "PURSUIT";
253     }
254     else if(mode == EVADE){
255        createAgents();
256        scenario = "EVADE";
257     }
258     else if(mode == AVOID_OBSTACLE){
259        createAgents();
260        createObstacle(obstacles);
261        scenario = "OBSTACLE AVOIDANCE";
262     }
263
264     view = graphics();
265     view.initGraphics(argv, argc, loop);
266 }
```

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

6.11.1.6 loop()

```
void loop ( )
```

Definition at line 98 of file main.cpp.

```
98              {
99     view.refreshScene();
100     //TODO: create scenario abstract class and inherit all scenarios from it, remove code below
101     for(auto it = agents.begin(); it < agents.end(); it++){
102        if(mode==FLOCK){
103           view.forceInScreen((*it));
104
105           pvector sep = behavior.separation(agents, *it);
106           sep.mul(1.5);
107           pvector ali = behavior.align(agents, *it);
108           ali.mul(4);
109           pvector coh = behavior.cohesion(agents, *it);
110           coh.mul(0.1);
111
112           (*it).force = sep + ali + coh;
113           (*it).desiredVelocity = (*it).force + (*it).velocity;
114           (*it).targetPoint = (*it).position + (*it).desiredVelocity;
115           (*it).arrive = true;
116        }
117
118        else if (mode == FOLLOW_MOUSE){
```

```
119             (*it).targetPoint = view.getMousePosition();
120             (*it).force  = behavior.seek(*it);
121             (*it).arrive = true;
122         }
123 
124         else if (mode == STAY_IN_FIELD){
125             view.drawWall(WALL, myColor);
126             (*it).force  = behavior.stayInArea(*it, WALL - DISTANCE);
127             (*it).force += behavior.separation(agents, *it);
128         }
129 
130         else if(mode == IN_FLOW_FIELD){
131             flow = flowField(pvector(GRAVITY));
132             (*it).force  = behavior.inFlowField(*it, flow);
133 
134             flow = flowField(pvector(WIND_WEST));
135             (*it).force += behavior.inFlowField(*it, flow);
136         }
137 
138         else if(mode == STAY_IN_PATH){
139             view.drawPath(way, myColor);
140             (*it).force  = behavior.stayInPath(*it, way);
141             (*it).force += behavior.separation(agents, *it);
142         }
143 
144         else if(mode == STAY_IN_PATH_2){
145             view.drawPath(way, myColor);
146             pvector seek = behavior.stayInPath_2(*it, way, view);
147             pvector sep  = behavior.separation(agents, *it);
148             sep.mul(5);
149             (*it).force = sep + seek;
150         }
151 
152         else if(mode == WANDER){//TODO: logic must be improved
153             (*it).force = behavior.wander(*it);
154         }
155 
156         else if(mode == FLEE){
157             (*it).force = behavior.flee((*it), view, view.getMousePosition());
158         }
159 
160         else if(mode == PURSUIT){
161             if((*it).name == "gazelle"){
162                 (*it).targetPoint = view.getMousePosition();
163                 (*it).force  = behavior.seek(*it);
164             }
165             else{//lion
166                 (*it).force  = behavior.pursuit(agents, *it, view);
167             }
168             (*it).arrive = true;
169         }
170 
171         else if(mode == EVADE){
172             if((*it).name == "lion"){
173                 (*it).targetPoint = view.getMousePosition();
174                 (*it).force  = behavior.seek(*it);
175                 (*it).arrive = true;
176             }
177             else{//gazelle
178                 (*it).force  = behavior.evade(agents, *it, view);
179             }
180         }
181 
182         else if(mode == AVOID_OBSTACLE){
183             for(auto it = obstacles.begin(); it < obstacles.end(); it++){
184                 point p = (*it).p;
185                 view.drawCircle(p, (*it).r);
186             }
187 
188             (*it).targetPoint = view.getMousePosition();
189             pvector seek  = behavior.seek(*it);
190             seek.mul(0.5);
191 
192             pvector avoid = behavior.avoid(obstacles, *it);
193             (*it).force = avoid + seek;
194             (*it).arrive = true;
195         }
196     }
197 
198     for(auto it = agents.begin(); it < agents.end(); it++){
199         (*it).updatePosition(mode, (*it).arrive);
200         view.drawAgent(*it, (*it).fillColor);
201     }
202 
203     view.drawText(scenario, point(-34, 32.25)); //TODO: magic numbers, define left corner
204 }
```

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

6.11.1.7 main()

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

Definition at line 268 of file main.cpp.

```
268                                 {
269     menu();
270     init(&argc, argv, loop);
271     return 0;
272 }
```

Here is the call graph for this function:

6.11.1.8 menu()

```
void menu ( )
```

Definition at line 28 of file main.cpp.

```
28              {
29     cout « "Follow Mouse       : 1" « endl;
30     cout « "Stay in Field      : 2" « endl;
31     cout « "In Flow Field      : 3" « endl;
32     cout « "Stay in Path       : 4" « endl;
33     cout « "Stay in Path 2     : 5" « endl;
34     cout « "FLOCK              : 6" « endl;
35     cout « "WANDER             : 7" « endl;
36     cout « "FLEE               : 8" « endl;
37     cout « "PURSUIT            : 9" « endl;
38     cout « "EVADE              : 10" « endl;
39     cout « "OBSTACLE AVOIDANCE : 11" « endl;
40     cin » mode;
41 }
```

Here is the caller graph for this function:

6.11.2 Variable Documentation

6.11.2.1 agents

```
vector<agent> agents
```

Definition at line 26 of file main.cpp.

6.11.2.2 behavior

```
steeringBehavior behavior
```

Definition at line 22 of file main.cpp.

6.11.2.3 flow

```
flowField flow
```

Definition at line 19 of file main.cpp.

6.11.2.4 mode

```
int mode
```

Definition at line 18 of file main.cpp.

6.11.2.5 myColor

```
color myColor
```

Definition at line 25 of file main.cpp.

6.11.2.6 obstacles

```
vector<obstacle> obstacles
```

Definition at line 24 of file main.cpp.

6.11.2.7 scenario

```
string scenario
```

Definition at line 23 of file main.cpp.

6.11.2.8 view

```
graphics view
```

Definition at line 20 of file main.cpp.

6.11.2.9 way

```
path way
```

Definition at line 21 of file main.cpp.

6.12 /home/user/Desktop/mm/autonomousSteeringAgents/README.md File Reference

6.13 /home/user/Desktop/mm/autonomousSteeringAgents/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:

6.14 /home/user/Desktop/mm/autonomousSteeringAgents/src/color.cpp File Reference

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

Include dependency graph for color.cpp:

6.15 /home/user/Desktop/mm/autonomousSteeringAgents/src/flowField.cpp File Reference

flowField class implementation

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

Include dependency graph for flowField.cpp:

6.15.1 Detailed Description

flowField class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

13.05.2021

6.16 /home/user/Desktop/mm/autonomousSteering↩ Agents/src/graphics.cpp File Reference

```
#include "graphics.h"
#include <GL/glut.h>
#include <iostream>
#include "math.h"
```

Include dependency graph for graphics.cpp:

6.17 /home/user/Desktop/mm/autonomousSteering↩ Agents/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:

6.17.1 Detailed Description

obstacle class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.18 /home/user/Desktop/mm/autonomousSteeringAgents/src/path.cpp File Reference

path class implementation

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

Include dependency graph for path.cpp:

6.18.1 Detailed Description

path class implementation

Author

Mehmet Rıza Öz - mehmetrizaoz@gmail.com

Date

12.05.2021

6.19 /home/user/Desktop/mm/autonomousSteeringAgents/src/point.cpp File Reference

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

Include dependency graph for point.cpp:

6.20 /home/user/Desktop/mm/autonomousSteering↩ Agents/src/pvector.cpp File Reference

```
#include "pvector.h"
#include "math.h"
#include "point.h"
#include <iostream>
#include <string>
```

Include dependency graph for pvector.cpp:

6.21 /home/user/Desktop/mm/autonomousSteering↩ Agents/src/random.cpp File Reference

```
#include "random.h"
#include <stdlib.h>
#include <iostream>
```

Include dependency graph for random.cpp:

6.22 /home/user/Desktop/mm/autonomousSteeringAgents/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:

6.23 /home/user/Desktop/mm/autonomousSteeringAgents/unit_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)

6.23.1 Macro Definition Documentation

6.23.1.1 BOOST_TEST_MODULE

```
#define BOOST_TEST_MODULE test_suites
```

Definition at line 1 of file test_suites.cpp.

6.23.2 Function Documentation

6.23.2.1 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:

6.23.2.2 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:

6.23.2.3 BOOST_AUTO_TEST_CASE() [3/12]

```
BOOST_AUTO_TEST_CASE (
              s1t3  )
```

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:

6.23.2.4 BOOST_AUTO_TEST_CASE() [4/12]

```
BOOST_AUTO_TEST_CASE (
              s1t4  )
```

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:

6.23.2.5 BOOST_AUTO_TEST_CASE() [5/12]

```
BOOST_AUTO_TEST_CASE (
              s1t5  )
```

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:

6.23.2.6 BOOST_AUTO_TEST_CASE() [6/12]

```
BOOST_AUTO_TEST_CASE (
              s1t6  )
```

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:

6.23.2.7 BOOST_AUTO_TEST_CASE() [7/12]

```
BOOST_AUTO_TEST_CASE (
              s1t7  )
```

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:

6.23.2.8 BOOST_AUTO_TEST_CASE() [8/12]

```
BOOST_AUTO_TEST_CASE (
              s1t8  )
```

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:

6.23.2.9 BOOST_AUTO_TEST_CASE() [9/12]

```
BOOST_AUTO_TEST_CASE (
              s1t9  )
```

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:

6.23.2.10 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:

6.23.2.11 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:

6.23.2.12 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

/home/user/Desktop/mm/autonomousSteeringAgents/README.md, 62
/home/user/Desktop/mm/autonomousSteeringAgents/include/agent.h, 47
/home/user/Desktop/mm/autonomousSteeringAgents/include/color.h, 47
/home/user/Desktop/mm/autonomousSteeringAgents/include/flowField.h, 48
/home/user/Desktop/mm/autonomousSteeringAgents/include/graphics.h, 49
/home/user/Desktop/mm/autonomousSteeringAgents/include/obstacle.h, 50
/home/user/Desktop/mm/autonomousSteeringAgents/include/path.h, 51
/home/user/Desktop/mm/autonomousSteeringAgents/include/point.h, 51
/home/user/Desktop/mm/autonomousSteeringAgents/include/pvector.h, 52
/home/user/Desktop/mm/autonomousSteeringAgents/include/random.h, 52
/home/user/Desktop/mm/autonomousSteeringAgents/include/steeringBehavior.h, 53
/home/user/Desktop/mm/autonomousSteeringAgents/main.cpp, 55
/home/user/Desktop/mm/autonomousSteeringAgents/src/agent.cpp, 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/color.cpp, 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/flowField.cpp, 62
/home/user/Desktop/mm/autonomousSteeringAgents/src/graphics.cpp, 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/obstacle.cpp, 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/path.cpp, 63
/home/user/Desktop/mm/autonomousSteeringAgents/src/point.cpp, 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/pvector.cpp, 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/random.cpp, 64
/home/user/Desktop/mm/autonomousSteeringAgents/src/steeringBehavior.cpp, 65
/home/user/Desktop/mm/autonomousSteeringAgents/unit_test/test_suites.cpp, 65
~agent
    agent, 10

acceleration
    agent, 11
add
    pvector, 34
addPoint
    path, 28
agent, 9
    agentID, 10
    acceleration, 11
    agents, 11
    arrive, 11
    calcDesiredVelocity, 11
    fillColor, 11
    force, 12
    id, 12
    mass, 12
    maxForce, 12
    maxSpeed, 12
    name, 12
    position, 13
    r, 13
    setFlowBehavior, 10
    steering, 13
    targetPoint, 13
    updatePosition, 10
    velocity, 13
agents
    main.cpp, 60
align
    steeringBehavior, 40
angleBetween
    pvector, 34
arrive
    agent, 11
avoid
    steeringBehavior, 40
AVOID_OBSTACLE
    steeringBehavior.h, 53

B
    color, 15
behavior
    main.cpp, 60
BLACK
    color.h, 48
BLUE
    color.h, 48
BOOST_AUTO_TEST_CASE
    test_suites.cpp, 66–68
BOOST_TEST_MODULE
    test_suites.cpp, 65

CIRCLE_DISTANCE
steeringBehavior.h, 53
CIRCLE_RADIUS
steeringBehavior.h, 53
cohesion
steeringBehavior, 41
color, 14
B, 15
color, 14
colors, 15
createColors, 15
G, 15
getColor, 15
R, 16
color.h
BLACK, 48
BLUE, 48
CYAN, 48
GREEN, 48
MAGENDA, 48
num, 47
RED, 48
WHITE, 48
YELLOW, 48
colors
color, 15
createAgents
main.cpp, 56
createColors
color, 15
createObstacle
main.cpp, 56
createPath_1
path, 28
createPath_2
path, 28
createRandomAgents
main.cpp, 56
createRandomArray
random, 39
createTroop
main.cpp, 57
CYAN
color.h, 48

desiredVelocity
agent, 11
div
point, 30
pvector, 35
dotProduct
pvector, 35
drawAgent
graphics, 19
drawCircle
graphics, 19
drawLine
graphics, 19
drawPath
graphics, 20
drawPoint
graphics, 20
drawText
graphics, 20
drawWall
graphics, 21

ESC
graphics.h, 50
EVADE
steeringBehavior.h, 54
evade
steeringBehavior, 41

fillColor
agent, 11
FLEE
steeringBehavior.h, 54
flee
steeringBehavior, 42
FLOCK
steeringBehavior.h, 54
flow
main.cpp, 60
flowField, 16
flowField, 16, 17
getField, 17
flowField.h
GRAVITY, 49
HEIGHT, 49
WIDTH, 49
WIND_WEST, 49
FOLLOW_MOUSE
steeringBehavior.h, 54
force
agent, 12
forceInScreen
graphics, 21

G
color, 15
getAngle
pvector, 35
getColor
color, 15
getField
flowField, 17
getMousePosition
graphics, 21
getNormalPoint
point, 31
graphics, 18
drawAgent, 19
drawCircle, 19
drawLine, 19
drawPath, 20
drawPoint, 20
drawText, 20

drawWall, 21
forceInScreen, 21
getMousePosition, 21
handleKeypress, 22
handleResize, 22
initGraphics, 22
mouseButton, 22
mouseMove, 23
refreshScene, 23
target_x, 24
target_y, 24
timerEvent, 23
graphics.h
ESC, 50
HEIGHT, 50
PI, 50
WIDTH, 50
GRAVITY
flowField.h, 49
GREEN
color.h, 48

handleKeypress
graphics, 22
handleResize
graphics, 22
HEIGHT
flowField.h, 49
graphics.h, 50

id
agent, 12
IN_FLOW_FIELD
steeringBehavior.h, 54
inFlowField
steeringBehavior, 42
init
main.cpp, 57
initGraphics
graphics, 22

limit
pvector, 35
loop
main.cpp, 58

MAGENDA
color.h, 48
magnitude
pvector, 36
main
main.cpp, 60
main.cpp
agents, 60
behavior, 60
createAgents, 56
createObstacle, 56
createRandomAgents, 56
createTroop, 57
flow, 60
init, 57
loop, 58
main, 60
menu, 60
mode, 61
myColor, 61
obstacles, 61
scenario, 61
view, 61
way, 61
mass
agent, 12
maxForce
agent, 12
maxSpeed
agent, 12
menu
main.cpp, 60
mode
main.cpp, 61
mouseButton
graphics, 22
mouseMove
graphics, 23
mul
point, 31
pvector, 36
myColor
main.cpp, 61

name
agent, 12
normalize
pvector, 36
num
color.h, 47

obstacle, 24
obstacle, 25
p, 25
r, 26
obstacles
main.cpp, 61
operator+
point, 31
pvector, 36, 37
operator+=
pvector, 37
operator-
point, 32
pvector, 37
operator==
point, 32
pvector, 38

p
obstacle, 25
path, 26

addPoint, 28
createPath_1, 28
createPath_2, 28
path, 27
points, 29
width, 29
PI
graphics.h, 50
pvector.h, 52
point, 29
div, 30
getNormalPoint, 31
mul, 31
operator+, 31
operator-, 32
operator==, 32
point, 30
print, 32
x, 32
y, 33
points
path, 29
position
agent, 13
print
point, 32
pvector, 38
PURSUIT
steeringBehavior.h, 54
pursuit
steeringBehavior, 42
pvector, 33
add, 34
angleBetween, 34
div, 35
dotProduct, 35
getAngle, 35
limit, 35
magnitude, 36
mul, 36
normalize, 36
operator+, 36, 37
operator+=, 37
operator-, 37
operator==, 38
print, 38
pvector, 34
x, 38
y, 38
pvector.h
PI, 52

R
color, 16
r
agent, 13
obstacle, 26
random, 39
createRandomArray, 39
RED
color.h, 48
refreshScene
graphics, 23

scenario
main.cpp, 61
seek
steeringBehavior, 43
separation
steeringBehavior, 43
setAngle
steeringBehavior, 43
setFeatures
agent, 10
STAY_IN_FIELD
steeringBehavior.h, 55
STAY_IN_PATH
steeringBehavior.h, 55
STAY_IN_PATH_2
steeringBehavior.h, 55
stayInArea
steeringBehavior, 44
stayInPath
steeringBehavior, 44
stayInPath_2
steeringBehavior, 44
steering
agent, 13
steeringBehavior, 40
align, 40
avoid, 40
cohesion, 41
evade, 41
flee, 42
inFlowField, 42
pursuit, 42
seek, 43
separation, 43
setAngle, 43
stayInArea, 44
stayInPath, 44
stayInPath_2, 44
wander, 45
steeringBehavior.h
AVOID_OBSTACLE, 53
CIRCLE_DISTANCE, 53
CIRCLE_RADIUS, 53
EVADE, 54
FLEE, 54
FLOCK, 54
FOLLOW_MOUSE, 54
IN_FLOW_FIELD, 54
PURSUIT, 54
STAY_IN_FIELD, 55
STAY_IN_PATH, 55
STAY_IN_PATH_2, 55
WANDER, 55

target_x
graphics, 24
target_y
graphics, 24
targetPoint
agent, 13
test_suites.cpp
BOOST_AUTO_TEST_CASE, 66–68
BOOST_TEST_MODULE, 65
timerEvent
graphics, 23

updatePosition
agent, 10

velocity
agent, 13
view
main.cpp, 61

WANDER
steeringBehavior.h, 55
wander
steeringBehavior, 45
way
main.cpp, 61
WHITE
color.h, 48
WIDTH
flowField.h, 49
graphics.h, 50
width
path, 29
WIND_WEST
flowField.h, 49

x
point, 32
pvector, 38

y
point, 33
pvector, 38
YELLOW
color.h, 48