RocketRobot

Generated by Doxygen 1.8.6

Sat May 9 2015 17:33:43

Contents

1	Sun	nmary																1
	1.1	Overvi	ew					 		 	 	 		 	 			1
		1.1.1	Installation	on				 		 	 	 		 	 			1
		1.1.2	User Inte	erfac	e			 		 	 	 		 	 			1
	1.2	Neural	network o	ptim	nizatio	on .		 		 	 	 		 	 			2
	1.3	Implen	nentation					 		 	 	 		 	 			2
		1.3.1	Object m	notio	n			 		 	 	 		 	 			2
		1.3.2	Graphics	s .				 		 	 	 		 	 			2
		1.3.3	Configur	atior	ı syst	em		 		 	 	 		 	 			2
	1.4	Licenc	e					 		 	 	 		 				3
2	Nam	nespace	Index															5
	2.1	Names	space List					 		 	 	 		 				5
3	Hier	archica	l Index															7
	3.1	Class	Hierarchy					 		 	 	 		 	 			7
4	Clas	ss Index	[9
	4.1	Class	List					 		 	 	 		 				9
5	File	Index																11
	5.1	File Lis	st					 		 	 	 		 				11
6	Nam	nespace	Docume	ntati	ion													13
	6.1	artist N	Namespac	e Re	eferen	ice		 		 	 	 		 	 			13
		6.1.1	Detailed	Des	cription	on		 		 	 	 		 	 			13
		6.1.2	Function	Doc	cumer	ntatio	on	 		 	 	 		 	 			13
			6.1.2.1	de	bugA	rrow		 		 	 	 		 	 			13
			6.1.2.2	dra	awLig	ht .		 		 	 	 		 	 			13
			6.1.2.3	dra	awOb	ject		 		 	 	 		 	 			13
			6.1.2.4	dra	awOb	stac	le	 		 	 	 		 				14
			6.1.2.5	dra	awRo	bot		 		 	 	 		 	 			14
			6.1.2.6	dra	awSei	nsor		 		 	 			 				14

iv CONTENTS

6.2	enviror	nment Nan	nespace Reference							
	6.2.1	Detailed	Detailed Description							
	6.2.2	Typedef I	Documentation							
		6.2.2.1	objectIterator							
	6.2.3	Function	Documentation							
		6.2.3.1	addObject							
		6.2.3.2	clear							
		6.2.3.3	getCollisionId							
		6.2.3.4	getCollisionId							
		6.2.3.5	getCollisionId							
		6.2.3.6	getHitableCollisionId							
		6.2.3.7	getHitableCollisionId							
		6.2.3.8	getHitableCollisionId							
		6.2.3.9	getNumObjects							
		6.2.3.10	getObject							
		6.2.3.11	getObjectsBegin							
		6.2.3.12	getObjectsEnd							
		6.2.3.13	isColliding							
		6.2.3.14	isColliding							
		6.2.3.15	isCollidingWithHitable							
		6.2.3.16	isCollidingWithHitable							
		6.2.3.17	isTouchingHitableObject							
		6.2.3.18	isTouchingHitableObject							
		6.2.3.19	isTouchingHitableObject							
		6.2.3.20	isTouchingObject							
		6.2.3.21	isTouchingObject							
		6.2.3.22	isTouchingObject							
		6.2.3.23	isTouchingWall							
		6.2.3.24	isTouchingWall							
		6.2.3.25	removeObject							
6.3	util Na	mespace F	Reference							
	6.3.1	Detailed	Description							
	6.3.2	Function	Documentation							
		6.3.2.1	addMovingLightSource							
		6.3.2.2	addNeuralNetworkRobotTarget							
		6.3.2.3	addObstacle							
		6.3.2.4	addRobot							
		6.3.2.5	addRobotTarget							
		6.3.2.6	addStationaryLightSource							
		6.3.2.7	advance							

CONTENTS

			6.3.2.8	copy	25
			6.3.2.9	display	26
			6.3.2.10	getNumLights	26
			6.3.2.11	getNumObstacles	26
			6.3.2.12	getNumRobotsTargets	26
			6.3.2.13	newColor	26
			6.3.2.14	open	26
			6.3.2.15	removeAllLightSource	26
			6.3.2.16	removeAllObstacle	26
			6.3.2.17	removeAllRobotTarget	26
			6.3.2.18	removeLightSource	27
			6.3.2.19	removeObstacle	27
			6.3.2.20	removeRobotTarget	27
			6.3.2.21	reset	27
			6.3.2.22	save	27
7	Clas	s Docui	mentation		29
	7.1			ss Reference	29
		7.1.1		Description	30
		7.1.2		tor & Destructor Documentation	30
			7.1.2.1	BaseGfxApp	30
			7.1.2.2	~BaseGfxApp	30
		7.1.3	Member	Function Documentation	30
			7.1.3.1	advance	30
			7.1.3.2	display	30
			7.1.3.3	glui	31
			7.1.3.4	gluiControl	31
			7.1.3.5	graphicsBegin	31
			7.1.3.6	graphicsTimer	31
			7.1.3.7	handle	31
			7.1.3.8	height	31
			7.1.3.9	keyboard	31
			7.1.3.10	keyboardSpecial	31
			7.1.3.11	keyboardSpecialUp	31
			7.1.3.12	keyboardUp	31
			7.1.3.13	leftMouseDown	31
			7.1.3.14	leftMouseUp	31
			7.1.3.15	middleMouseDown	31
			7.1.3.16	middleMouseUp	31
			7.1.3.17	mouseDragged	31

vi CONTENTS

		7.1.3.18	mouseMoved	32
		7.1.3.19	reshape	32
		7.1.3.20	rightMouseDown	32
		7.1.3.21	rightMouseUp	32
		7.1.3.22	runMainLoop	32
		7.1.3.23	s_draw	32
		7.1.3.24	s_gluicallback	32
		7.1.3.25	s_keyboard	32
		7.1.3.26	s_keyboardspecial	32
		7.1.3.27	s_keyboardspecialup	32
		7.1.3.28	s_keyboardup	32
		7.1.3.29	s_mousebtn	32
		7.1.3.30	s_mousemotion	32
		7.1.3.31	s_reshape	32
		7.1.3.32	setCaption	32
		7.1.3.33	width	32
	7.1.4	Member	Data Documentation	32
		7.1.4.1	m_drag	32
		7.1.4.2	m_glui	32
		7.1.4.3	m_glutWindowHandle	32
		7.1.4.4	m_height	32
		7.1.4.5	m_width	32
		7.1.4.6	s_currentApp	32
		7.1.4.7	s_glutInitialized	33
7.2	Color S	Struct Refe	erence	33
	7.2.1	Detailed	Description	33
	7.2.2	Construc	ctor & Destructor Documentation	33
		7.2.2.1	Color	33
		7.2.2.2	Color	33
		7.2.2.3	Color	34
		7.2.2.4	Color	34
	7.2.3	Member	Function Documentation	34
		7.2.3.1	isSimilar	34
		7.2.3.2	operator!=	34
		7.2.3.3	operator<<	35
		7.2.3.4	operator==	35
	7.2.4	Member	Data Documentation	35
		7.2.4.1	blue	35
		7.2.4.2	green	35
		7.2.4.3	red	35

CONTENTS vii

7.3	Compl	exRobot C	lass Reference	35				
	7.3.1	Detailed Description						
	7.3.2	Construc	tor & Destructor Documentation	36				
		7.3.2.1	ComplexRobot	36				
		7.3.2.2	ComplexRobot	36				
		7.3.2.3	ComplexRobot	36				
		7.3.2.4	\sim ComplexRobot	37				
	7.3.3	Member	Function Documentation	37				
		7.3.3.1	getLeftSpeed	37				
		7.3.3.2	getRightSpeed	37				
	7.3.4	Member	Data Documentation	37				
		7.3.4.1	defaultSpeed	37				
		7.3.4.2	enableLightSensors	37				
		7.3.4.3	enableObstacleSensors	37				
		7.3.4.4	enableRobotSensors	37				
		7.3.4.5	enableTargetSensors	37				
		7.3.4.6	lightSensorScale	37				
		7.3.4.7	lightSensorsCrossed	37				
		7.3.4.8	obstacleSensorScale	37				
		7.3.4.9	obstacleSensorsCrossed	37				
		7.3.4.10	robotSensorScale	37				
		7.3.4.11	robotSensorsCrossed	37				
		7.3.4.12	targetSensorScale	37				
		7.3.4.13	targetSensorsCrossed	37				
7.4	Config	uration Cla	ass Reference	37				
	7.4.1	Member	Function Documentation	38				
		7.4.1.1	get	38				
		7.4.1.2	getBoolConfig	38				
		7.4.1.3	getCharConfig	38				
		7.4.1.4	getFloatConfig	38				
		7.4.1.5	getIntConfig	39				
		7.4.1.6	getStringConfig	39				
		7.4.1.7	initConfig	39				
		7.4.1.8	initConfig	39				
		7.4.1.9	refresh	39				
7.5	config\	/alue Strud	ct Reference	39				
	7.5.1	Detailed	Description	40				
	7.5.2	Member	Data Documentation	40				
		7.5.2.1	"@1	40				
		7.5.2.2	boolVal	40				

viii CONTENTS

		7.5.2.3	charVal	40
		7.5.2.4	floatVal	40
		7.5.2.5	intVal	40
		7.5.2.6	stringVal	40
		7.5.2.7	type	40
7.6	LightSo	ource Clas	s Reference	40
	7.6.1	Detailed I	Description	41
	7.6.2	Construct	tor & Destructor Documentation	41
		7.6.2.1	LightSource	41
		7.6.2.2	LightSource	41
		7.6.2.3	LightSource	42
		7.6.2.4	~LightSource	42
	7.6.3	Member I	Function Documentation	42
		7.6.3.1	display	42
		7.6.3.2	handleCollision	42
7.7	Location	on Struct R	Reference	43
	7.7.1	Detailed I	Description	43
	7.7.2	Construct	tor & Destructor Documentation	43
		7.7.2.1	Location	43
		7.7.2.2	Location	43
	7.7.3	Member F	Function Documentation	43
		7.7.3.1	operator==	43
	7.7.4	Member I	Data Documentation	43
		7.7.4.1	x	43
		7.7.4.2	y	43
7.8	Neural	Network Cl	lass Reference	43
	7.8.1	Detailed I	Description	44
	7.8.2	Construct	tor & Destructor Documentation	44
		7.8.2.1	NeuralNetwork	44
		7.8.2.2	NeuralNetwork	44
	7.8.3	Member I	Function Documentation	44
		7.8.3.1	combine	44
		7.8.3.2	compute	44
		7.8.3.3	load	45
		7.8.3.4	mutate	46
		7.8.3.5	write	46
7.9	Neural	NetworkRo	obot Class Reference	46
	7.9.1	Detailed I	Description	47
	7.9.2	Construct	tor & Destructor Documentation	47
		7.9.2.1	NeuralNetworkRobot	47

CONTENTS

		7.9.2.2	NeuralNetworkRobot	. 47
		7.9.2.3	\sim NeuralNetworkRobot	. 47
	7.9.3	Member F	Function Documentation	. 47
		7.9.3.1	getLeftSpeed	. 47
		7.9.3.2	getRightSpeed	. 47
	7.9.4	Member D	Data Documentation	. 47
		7.9.4.1	filename	. 47
7.10	NoOpe	nLocationE	Exception Class Reference	. 48
	7.10.1	Detailed D	Description	. 48
	7.10.2	Construct	tor & Destructor Documentation	. 48
		7.10.2.1	NoOpenLocationException	. 48
7.11	Objectl	terator Cla	ss Reference	. 48
	7.11.1	Detailed D	Description	. 49
	7.11.2	Construct	tor & Destructor Documentation	. 49
		7.11.2.1	ObjectIterator	. 49
		7.11.2.2	ObjectIterator	. 49
		7.11.2.3	~ObjectIterator	. 49
	7.11.3	Member F	Function Documentation	. 49
		7.11.3.1	operator!=	. 49
		7.11.3.2	operator*	. 49
		7.11.3.3	operator++	. 49
		7.11.3.4	operator++	. 49
		7.11.3.5	operator<	. 49
		7.11.3.6	operator==	. 49
		7.11.3.7	operator>	. 49
	7.11.4	Friends A	nd Related Function Documentation	. 49
		7.11.4.1	environment::getObjectsBegin	. 49
		7.11.4.2	environment::getObjectsEnd	. 49
7.12	Obstac	le Class Re	eference	. 50
	7.12.1	Detailed D	Description	. 50
	7.12.2	Construct	tor & Destructor Documentation	. 50
		7.12.2.1	Obstacle	. 50
		7.12.2.2	Obstacle	. 50
		7.12.2.3	Obstacle	. 51
		7.12.2.4	~Obstacle	. 51
	7.12.3	Member F	Function Documentation	. 51
		7.12.3.1	handleCollision	. 51
7.13	Optimiz	eSimulatio	on Class Reference	. 52
	7.13.1	Detailed D	Description	. 52
	7.13.2	Construct	tor & Destructor Documentation	. 52

CONTENTS

	7.13.2.1	OptimizeSimulation	 52
	7.13.2.2	\sim OptimizeSimulation	 52
7.13.3	Member	Function Documentation	 52
	7.13.3.1	getPerformance	 52
	7.13.3.2	getPerformanceMaze	 52
	7.13.3.3	getPerformanceObstacles	 52
	7.13.3.4	getPerformanceRandomRepeated	 52
	7.13.3.5	runMainLoop	 52
7.14 Physic	alObject C	Class Reference	 53
7.14.1	Detailed	Description	 54
7.14.2	Construc	ctor & Destructor Documentation	 54
	7.14.2.1	PhysicalObject	 54
	7.14.2.2	PhysicalObject	 54
	7.14.2.3	PhysicalObject	 54
	7.14.2.4	~PhysicalObject	 54
7.14.3	Member	Function Documentation	 54
	7.14.3.1	display	 54
	7.14.3.2	findOpenLocation	 54
	7.14.3.3	forceSetPosition	 54
	7.14.3.4	forceTranslate	 55
	7.14.3.5	getColor	 56
	7.14.3.6	getld	 56
	7.14.3.7	getLocation	 56
	7.14.3.8	getOrientation	 56
	7.14.3.9	getRadius	 56
	7.14.3.10	0 getSpeed	 56
	7.14.3.11	1 getXPosition	 57
	7.14.3.12	2 getYPosition	 57
	7.14.3.13	3 handleCollision	 57
	7.14.3.14	4 hasEqualPosition	 57
	7.14.3.15	5 pointTo	 57
	7.14.3.16	6 reorient	 58
	7.14.3.17	7 rotate	 58
	7.14.3.18	8 setColor	 58
	7.14.3.19	9 setLocation	 58
	7.14.3.20	0 setOrientation	 58
	7.14.3.21	1 setPosition	 59
	7.14.3.22	2 setRadius	 59
	7.14.3.23	3 setSpeed	 59
	7.14.3.24	4 translate	 59

CONTENTS xi

		7.14.3.25	update	59
		7.14.3.26	updateMembers	59
		7.14.3.27	updatePosition	60
	7.14.4	Member I	Data Documentation	60
		7.14.4.1	isHitable	60
		7.14.4.2	objectType	60
7.15	Robot (Class Refe	erence	60
	7.15.1	Detailed I	Description	61
	7.15.2	Construct	tor & Destructor Documentation	61
		7.15.2.1	Robot	61
		7.15.2.2	Robot	61
		7.15.2.3	\sim Robot	61
	7.15.3	Member I	Function Documentation	62
		7.15.3.1	display	62
		7.15.3.2	getLeftSpeed	62
		7.15.3.3	getLineColor	62
		7.15.3.4	getRightSpeed	62
		7.15.3.5	getTarget	62
		7.15.3.6	handleCollision	62
		7.15.3.7	setLineColor	62
		7.15.3.8	setTarget	63
		7.15.3.9	update	63
		7.15.3.10	updateMembers	63
	7.15.4	Member I	Data Documentation	63
		7.15.4.1	robotType	63
7.16	Sensor	Class Ref	erence	63
	7.16.1	Construct	tor & Destructor Documentation	64
		7.16.1.1	Sensor	64
		7.16.1.2	Sensor	64
	7.16.2	Member I	Function Documentation	64
		7.16.2.1	display	64
		7.16.2.2	getOrientation	64
		7.16.2.3	getPosition	65
		7.16.2.4	getTypeDetected	65
		7.16.2.5	getViewAngle	65
		7.16.2.6	getXPosition	65
		7.16.2.7	getYPosition	65
		7.16.2.8	sense	65
		7.16.2.9	setOrientation	66
		7.16.2.10	setPosition	67

xii CONTENTS

		7.16.2.11 setPosition	67
		7.16.2.12 setTypeDetected	67
		7.16.2.13 setViewAngle	67
		7.16.2.14 updatePosition	67
7.17	Simple	Robot Class Reference	68
	7.17.1	Detailed Description	68
	7.17.2	Constructor & Destructor Documentation	68
		7.17.2.1 SimpleRobot	68
		7.17.2.2 SimpleRobot	68
		7.17.2.3 ~SimpleRobot	68
	7.17.3	Member Function Documentation	68
		7.17.3.1 getLeftSpeed	68
		7.17.3.2 getRightSpeed	69
7.18	Simulat	tion Class Reference	69
	7.18.1	Detailed Description	71
	7.18.2	Member Enumeration Documentation	71
		7.18.2.1 UIControlType	71
	7.18.3	Constructor & Destructor Documentation	71
		7.18.3.1 Simulation	71
		7.18.3.2 ~Simulation	71
	7.18.4	Member Function Documentation	71
		7.18.4.1 advance	71
		7.18.4.2 display	71
		7.18.4.3 gluiControl	72
		7.18.4.4 initObjects	72
		7.18.4.5 keyboard	72
		7.18.4.6 keyboardSpecial	72
		7.18.4.7 leftMouseDown	72
		7.18.4.8 leftMouseUp	72
		7.18.4.9 middleMouseDown	72
		7.18.4.10 mouseDragged	72
		7.18.4.11 pause	72
		7.18.4.12 reinit	72
		7.18.4.13 resume	72
		7.18.4.14 s_addMovingLightSource	72
		7.18.4.15 s_addObstacle	72
		7.18.4.16 s_addRobot	73
		7.18.4.17 s_addRobotTarget	73
		7.18.4.18 s_addStationaryLightSource	73
		7.18.4.19 s_neuralNetworkFileChanged	73

CONTENTS xiii

		7.18.4.20 s_open	73
		7.18.4.21 s_pause	73
		7.18.4.22 s_refreshConfiguration	73
		7.18.4.23 s_reinit	73
		7.18.4.24 s_removeAllLightSource	73
		7.18.4.25 s_removeAllObstacle	73
		7.18.4.26 s_removeAllRobotTarget	73
		7.18.4.27 s_removeLightSource	73
		7.18.4.28 s_removeObstacle	73
		7.18.4.29 s_removeRobotTarget	73
		7.18.4.30 s_resume	73
		7.18.4.31 s_save	73
		7.18.4.32 s_start	73
		7.18.4.33 start	73
		7.18.4.34 tryAddMovingLightSource	73
		7.18.4.35 tryAddObstacle	73
		7.18.4.36 tryAddRobot	73
		7.18.4.37 tryAddRobotTarget	73
		7.18.4.38 tryAddStationaryLightSource	74
		7.18.4.39 tryOpen	74
		7.18.4.40 tryRemoveLightSource	74
		7.18.4.41 tryRemoveObstacle	74
		7.18.4.42 tryRemoveRobotTarget	74
		7.18.4.43 trySave	74
7.19	Target	Class Reference	74
	7.19.1	Detailed Description	75
	7.19.2	Constructor & Destructor Documentation	75
		7.19.2.1 Target	75
		7.19.2.2 Target	75
		7.19.2.3 Target	75
		7.19.2.4 Target	75
		7.19.2.5 ~Target	75
	7.19.3	Member Function Documentation	75
		7.19.3.1 handleCollision	75
Eile.	D		77
8.1		entation pp File Reference	77 77
0.1	8.1.1	Macro Definition Documentation	77
	0.1.1	8.1.1.1 _USE_MATH_DEFINES	77
8.2	artiet h	File Reference	77
0.2	ฉเ แจเ.ก	THE RESERVE	11

8

XIV

	8.2.1	Detailed Description	78
8.3	BaseG	App.cpp File Reference	78
	8.3.1	Detailed Description	78
8.4	BaseG	App.h File Reference	78
	8.4.1	Detailed Description	79
8.5	Color.c	p File Reference	79
	8.5.1	Detailed Description	79
8.6	Color.h	File Reference	79
	8.6.1	Detailed Description	80
	8.6.2	Macro Definition Documentation	80
		8.6.2.1 GET_COLOR	80
	8.6.3	Typedef Documentation	80
		8.6.3.1 Color	80
8.7	Comple	Robot.cpp File Reference	80
	8.7.1	Detailed Description	80
8.8	Comple	Robot.h File Reference	80
	8.8.1	Detailed Description	81
8.9	configu	ation.cpp File Reference	81
	8.9.1	Detailed Description	82
	8.9.2	Typedef Documentation	82
		8.9.2.1 ConfigTable	82
	8.9.3	Function Documentation	82
		8.9.3.1 expandVar	82
		8.9.3.2 isAllWhitespace	82
		8.9.3.3 loadConfig	82
		8.9.3.4 mergeConfigTables	82
		8.9.3.5 parseValue	82
8.10	configu	ation.h File Reference	82
	8.10.1	Detailed Description	83
	8.10.2	Macro Definition Documentation	83
		8.10.2.1 GET_BOOL	83
		8.10.2.2 GET_CHAR	83
		8.10.2.3 GET_FLOAT	83
		8.10.2.4 GET_INT	83
		8.10.2.5 GET_STRING	83
8.11	environ	nent.cpp File Reference	84
	8.11.1	Function Documentation	84
		8.11.1.1 isTouching	84
8.12	environ	nent.h File Reference	84
8.13	LightSo	urce.cpp File Reference	85

CONTENTS xv

	8.13.1 Detailed Description	85
8.14	LightSource.h File Reference	86
	8.14.1 Detailed Description	86
8.15	Location.h File Reference	86
	8.15.1 Typedef Documentation	86
	8.15.1.1 Location	86
8.16	main.cpp File Reference	86
	8.16.1 Detailed Description	87
	8.16.2 Macro Definition Documentation	87
	8.16.2.1 DEFAULT_CONFIG	87
	8.16.3 Function Documentation	87
	8.16.3.1 main	87
8.17	mainpage.h File Reference	87
8.18	NeuralNetwork.cpp File Reference	87
	8.18.1 Detailed Description	87
8.19	NeuralNetwork.h File Reference	88
	8.19.1 Detailed Description	88
8.20	NeuralNetworkRobot.cpp File Reference	88
	8.20.1 Detailed Description	88
8.21	NeuralNetworkRobot.h File Reference	88
	8.21.1 Detailed Description	89
8.22	Obstacle.cpp File Reference	89
	8.22.1 Detailed Description	89
8.23	Obstacle.h File Reference	89
	8.23.1 Detailed Description	90
8.24	OptimizeSimulation.cpp File Reference	90
8.25	OptimizeSimulation.h File Reference	90
	8.25.1 Detailed Description	91
8.26	PhysicalObject.cpp File Reference	91
	8.26.1 Detailed Description	91
	8.26.2 Macro Definition Documentation	91
	8.26.2.1 _USE_MATH_DEFINES	91
8.27	PhysicalObject.h File Reference	91
	8.27.1 Detailed Description	92
	8.27.2 Enumeration Type Documentation	92
	8.27.2.1 ObjectType	92
8.28	Robot.cpp File Reference	92
	8.28.1 Detailed Description	92
8.29	Robot.h File Reference	93
	8.29.1 Detailed Description	93

xvi CONTENTS

	8.29.2 Enumeration Type Documentation	93
	8.29.2.1 RobotType	93
8.30	Sensor.cpp File Reference	93
	8.30.1 Detailed Description	94
	8.30.2 Macro Definition Documentation	94
	8.30.2.1 _USE_MATH_DEFINES	94
8.31	Sensor.h File Reference	94
	8.31.1 Detailed Description	94
8.32	SimpleRobot.cpp File Reference	94
	8.32.1 Detailed Description	94
8.33	SimpleRobot.h File Reference	95
	8.33.1 Detailed Description	95
8.34	Simulation.cpp File Reference	95
	8.34.1 Detailed Description	96
	8.34.2 Function Documentation	96
	8.34.2.1 ftos	96
8.35	Simulation.h File Reference	96
	8.35.1 Detailed Description	96
8.36	Target.cpp File Reference	96
	8.36.1 Detailed Description	96
8.37	Target.h File Reference	97
	8.37.1 Detailed Description	97
8.38	util.cpp File Reference	97
	8.38.1 Detailed Description	98
	8.38.2 Variable Documentation	98
	8.38.2.1 colorNum	98
	8.38.2.2 lights	98
	8.38.2.3 objectsMutex	98
	8.38.2.4 obstacles	98
	8.38.2.5 robots	98
	8.38.2.6 targets	98
8.39	util.h File Reference	98
	8.39.1 Detailed Description	99

Index

100

Chapter 1

Summary

1.1 Overview

RocketRobot is a multi-featured robot simulator. It simulates robots, targets, and obstacles in an environment, with the goal of robots seeking and finding their targets. When a robot finds its target, both disappear from the simulation. In addition, stimuli or light sources can be placed in the environment. There are several types of robots. Simple robots have sensors that only seek the target, and ignore obstacles. Complex robots have sensors for lights, robots, obstacles, and the target. Note that obstacle sensors include the wall as an 'obstacle', straight ahead of the sensor. For both robot types, the wheels speeds are controlled directly by the sensor readings. Complex robots can have each type of sensor scaled or connected in any pattern, this is controllable from the 'New complex robot settings' tab in the interface. Neural network robots use a feed-forward neural network for control. The network takes the target, robot, and obstacle settings as inputs, and outputs the wheel speeds. The network is loaded from a file, specified in the 'New neural network robot settings' tab in the interface.

1.1.1 Installation

You can execute the 'gorobot' executable directly from the src or bin folder, but it is easier to directly install it. This allows you to run it as 'rocketrobot <optional filename>="">' from any folder. To install on a Linux computer, simply run the install script from the main folder. This soft-links a script to your \sim /bin folder. This requires that you have a \sim /bin folder on your path, if not you can create one and add it to your .bashrc or .cshrc. To uninstall, you can simply run the uninstall script.

1.1.2 User Interface

The RocketRobot user interface is fairly simple. The control panel allows you to start, stop, pause, resume, and quit the simulation. The reset button reverts to the previous time it was started if the simulation was opened from a file, or randomly generates a new simulation if the current simulation is random. The 'Refresh settings' button reloads the configuration files (described later). The open/save tab also lets you open or save the current state. You can also provide a file to open as the command-line argument. There are several examples simulations for this in the examples folder.

The user inteface is fairly straightforward for adding and removing objects. There are several control tabs from which the number of obstacles, robots, and lights can be controlled. Objects are added and removed in a FILO manner from the interface. Targets are always created with a robot, but not all robots need to have a target. Decreasing the number of robots through the control panel deletes the target paired with the removed robot.

For more information about an object (location, speed, orientation, radius), you can click on an object. You can also drag and drop objects to move them, or drag them off the screen to delete them. The radius off the selected object can be changed with the up and down arrow keys, the orientation with the left and right keys, and the speed with the + and - keys. You can also middle-click to paste a copy of the selected object. Note that copying a robot creates a new robot with the same target (if any.) You cannot copy targets.

2 Summary

1.2 Neural network optimization

A major portion of the project is the neural network-driven robot, and the associated optimization algorithims. To optimize the neural network robot, a seperate simulation class, OptimizeSimulation, was created. Optimization employs a genetic algorithm, in which a pool of possible networks is maintained, along with their performance. The performances are calculated by performing a large number of runs of (consistant) random simulations, and summing the total time for all robots to find all targets. In addition, several spesfic test cases are run for each network to test some interesting cases (e.g. navigating a simple maze, etc.) Pairs of networks are randomly selected from the pool (with one network from a better subset of the pool.) These are combined by taking half the weights from each and merging them into a new network, then making more random tweaks. This is then tested and added to the pool if it is better than the worst performance in the pool. The best performance in the pool is the optimal network.

To run the optimization, run ./optimize from the scripts folder. The temporary and intermediate files get saved to runtime/neuralnetwork. This includes the optimization log, pool performances, optimial network, pool networks, and several others.

1.3 Implementation

The various objects are all subclasses of PhysicalObject, which handles basic behaviors and attributes. This class is virtual, so that spesific bevhaviors for collisions, etc are handled individually, and can optionally call back to the default handlers in PhysicalObject. Each type of robot is a subclass of Robot, which is also virtual, as it has virtual functions to get new wheel speeds given the sensor readings. Everything else, including reading from sensors and collision behavior, is handled directly by the robot class.

There is also an environment namespace with which all objects regester, and are automaticly removed when they are destructed. An iterator over all objects can be requested from the environment, and they can also be accessed by id. The environment also contains functions for detecting collisions between walls and objects. A util namespace provides an easy way of adding and removing objects, managing stacks so that objects are added and removed in FILO order. It also contains functionality to open and save the current state to and from files.

The Simulation class handles the user interface, providing a wrapper for the functions in util. It also acts as a driver, requesting updates and redraws from all objects.

1.3.1 Object motion

Object movement is handled by PhysicalObject. It has an updateMembers function which advances all objects by a distance caluclated from the speed, after calling the virtual update function. The robot class update sets the position and speed from the calculated wheel speeds. The updateMembers function is called for each object from advance in util, which in turn is called from advance in Simulation which is triggered by a glut timer function.

1.3.2 Graphics

A graphical display of the simulation is rendered by OpenGL. Most of the graphics functionallity is in the artist namespace, which has simple functions for drawing circles and lines, which are used by more complex functions for different kinds of objects. There is a virtual function in PhysicalObject to render each object as a circle, and this is overridden by Robot and LightSource. These are called by simulation similarly to updatePosition.

1.3.3 Configuration system

Most settings are loaded through the configuration system, see configuration.h for more details. This allows settings to be stored externally, and avoids needing to recompile often when making small changes. Configuration values can also be set from the command line.

1.4 Licence 3

1.4 Licence

This project has been developed by Lucas Kramer, Carl Bahn, Himawan Go, and Xi Zhang. It has been extended by Lucas Kramer to add neural network-controlled robots and made the interface more useable. Copyright (C) 2015 Lucas Kramer, Carl Bahn, Himawan Go, and Xi Zhang

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

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

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

Summary

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

artist		
	Graphical commands	13
environn	nent	
411	Environment namespace, handles all the objects as a group. Also manages object ids and collision detection	14
util	Util namespace, contains helper functions to add and remove robots	22

6 Namespace Index

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

BaseGfxApp	29
Simulation	. 69
Color	33
Configuration	37
configValue	39
Location	43
NeuralNetwork	43
ObjectIterator	48
OptimizeSimulation	52
PhysicalObject	53
LightSource	. 40
Obstacle	. 50
Robot	. 60
ComplexRobot	. 35
NeuralNetworkRobot	. 46
SimpleRobot	. 68
Target	. 74
runtime_error	
NoOpenLocationException	. 48
Sensor	63

8 **Hierarchical Index**

Chapter 4

Class Index

4.1 Class List

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

29
33
35
37
39
40
43
43
46
40
48
40
48
50
50
52
52
53
50
60
63
00
68
69
74

10 Class Index

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief de	scriptions:
---	-------------

autiat and	77
artist.cpp	 . 77
A namespace for OpenGL graphical functions	77
BaseGfxApp.cpp	
Implementation of BaseGfxApp	 . 78
BaseGfxApp.h	
The basic application class for CSci-3081 project. Uses GLUT and GLUI and wraps the nice C++ interface	
Color.cpp	
Utility functions for handling color	 . 79
Color.h	
Definintions for representing any hexedecimal color	 . 79
ComplexRobot.cpp	
The representation of a robot within the simulation	 . 80
ComplexRobot.h	
The representation of a robot within the simulation	 . 80
configuration.cpp	
Implementation for configuration file loader	 . 81
configuration.h	
Interface definitions for configuration file loader	
environment.cpp	
environment.h	 . 84
LightSource.cpp	
The representation of a target in the simulation	 . 85
LightSource.h	
The representation of an obstacle in the simulation	
Location.h	 . 86
main.cpp	
The main function to execute the simulation program	
mainpage.h	 . 87
NeuralNetwork.cpp	
A feed-forward neural network class	 . 87
NeuralNetwork.h	
A feed-forward neural network class	 . 88
NeuralNetworkRobot.cpp	
The representation of a neural network robot within the simulation	 . 88
NeuralNetworkRobot.h	
The representation of a neural network robot	88

12 File Index

Obstacle.cpp	
The representation of an obstacle in the simulation	. 89
Obstacle.h	
The representation of an obstacle in the simulation	. 89
OptimizeSimulation.cpp	. 90
OptimizeSimulation.h	
Robot simultaion for optimizing the NeuralNetworkRobot	. 90
PhysicalObject.cpp	
The representation of any object within the simulation	. 91
PhysicalObject.h	
The representation of any object within the simulation	. 91
Robot.cpp	
The representation of robot within the simulation	. 92
Robot.h	
The representation of robot within the simulation	. 93
Sensor.cpp	
A sensor that detects certain objects in the environment	. 93
Sensor.h	
A sensor that detects certain objects in the environment	. 94
SimpleRobot.cpp	
The representation of robot within the simulation	. 94
SimpleRobot.h	
The representation of a simple robot	. 95
Simulation.cpp	
Implementation for the main application class of the robot simulation	. 95
Simulation.h	
Main application class for the robot simulation	. 96
Target.cpp	
The representation of a target in the simulation	. 96
Target.h	
The representation of an obstacle in the simulation	. 97
util.cpp	
Implmentation of functions for the simulation	. 97
util.h	
Helper functions for the simulation	. 98

Chapter 6

Namespace Documentation

6.1 artist Namespace Reference

graphical commands

Functions

- void drawObject (Location loc, int radius, Color color)
- void debugArrow (Location loc, int orientation)
- void drawLight (Location loc, int radius, Color color)
- void drawObstacle (Location loc, int radius)
- · void drawSensor (Location loc, int orientation, int angle, float intensity)
- · void drawRobot (Location loc, int radius, int orientation, Color color, Color lineColor)

6.1.1 Detailed Description

graphical commands

Author

Carl artist functions provide a contained way to display graphics without putting OpenGL graphics in each file

6.1.2 Function Documentation

6.1.2.1 void artist::debugArrow (Location loc, int orientation)

6.1.2.2 void artist::drawLight (Location loc, int radius, Color color)

Draws a LightSource

Parameters

loc	absolute Location to draw center of light
radius	radius of 'bulb' of light

6.1.2.3 void artist::drawObject (Location loc, int radius, Color color)

Author

Lucas Kramer Draws the default Object, which is a circle

Parameters

loc	absolute Location to draw center of circle
radius	radius of circle
color	the color of the circle

6.1.2.4 void artist::drawObstacle (Location loc, int radius)

Draws an obstacle

Parameters

loc	absolute Location to draw center of Obstacle
radius	radius of Obstacle

6.1.2.5 void artist::drawRobot (Location loc, int radius, int orientation, Color color, Color lineColor)

Draws a Robot

Parameters

loc	absolute Location of center of robot
orientation	absolute direction for robot to face
color	the color of the robot's body
lineColor	the color of the robot's direction line

6.1.2.6 void artist::drawSensor (Location loc, int orientation, int angle, float intensity)

Draws a sensor

Parameters

loc	absolute Location to draw center of Sensor
orientation	absolute direction of sensor
angle	number of degrees that sensor senses in, with orientation in center
intensity	redness of sensor as value between 0 and 1, meant to indicate amount of light detected

6.2 environment Namespace Reference

environment namespace, handles all the objects as a group. Also manages object ids and collision detection

Typedefs

• typedef ObjectIterator objectIterator

Functions

• int addObject (PhysicalObject *object)

Adds an object to the environment.

void removeObject (int id)

Removes an object from the environment.

• void clear ()

Removes all objects from the environment and resets the id counter.

unsigned getNumObjects ()

Gets the number of objects in environment.

PhysicalObject * getObject (int id)

Finds an object from the environment.

objectIterator getObjectsBegin ()

Gets an iterator to the beginning of the objects.

• objectIterator getObjectsEnd ()

Gets an iterator to the end of the objects.

- bool isTouchingWall (Location I, int r)
- bool isTouchingWall (int id)
- bool isTouchingObject (Location I, int r, int id)
- bool isTouchingHitableObject (Location I, int r, int id)
- bool isTouchingObject (Location I, int r)
- bool isTouchingHitableObject (Location I, int r)
- bool isTouchingObject (int id)
- bool isTouchingHitableObject (int id)
- bool isColliding (Location I, int r)
- bool isColliding (int id)
- bool isCollidingWithHitable (Location I, int r)
- bool isCollidingWithHitable (int id)
- int getCollisionId (Location I, int r, int id)
- int getHitableCollisionId (Location I, int r, int id)
- int getCollisionId (Location I, int r)
- int getHitableCollisionId (Location I, int r)
- int getCollisionId (int id)
- int getHitableCollisionId (int id)

6.2.1 Detailed Description

environment namespace, handles all the objects as a group. Also manages object ids and collision detection

6.2.2 Typedef Documentation

6.2.2.1 typedef ObjectIterator environment::objectIterator

An object iterator

6.2.3 Function Documentation

6.2.3.1 int environment::addObject (PhysicalObject * object)

Adds an object to the environment.

Parameters

object	The object to add

Returns

The assigned id of the object

6.2.3.2 void environment::clear ()

Removes all objects from the environment and resets the id counter.

6.2.3.3 int environment::getCollisionId (Location I, int r, int id)

Determines what object is being hit by another object

Parameters

1	The Location of the object
r	The radius of the object
id	The id of the object to exclude

Returns

The id of one object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.4 int environment::getCollisionId (Location I, int r)

Determines what object is being hit by another object

Parameters

The Location of the object
The radius of the object

Returns

The id of one object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.5 int environment::getCollisionId (int id)

Determines what object is being hit by another object

id	The id of the object
Id	The id of the object
	,

Returns

The id of one object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.6 int environment::getHitableCollisionId (Location I, int r, int id)

Determines what hitable object is being hit by another object

Parameters

1	The Location of the object
r	The radius of the object
id	The id of the object to exclude

Returns

The id of one hitable object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.7 int environment::getHitableCollisionId (Location l, int r)

Determines what hitable object is being hit by another object

Parameters

1	The Location of the object
r	The radius of the object

Returns

The id of one hitable object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.8 int environment::getHitableCollisionId (int id)

Determines what hitable object is being hit by another object

id The id of the object

Returns

The id of one hitable object that is being hit, or -1 if it is hiting nothing or a wall

See Also

isColliding

6.2.3.9 unsigned environment::getNumObjects ()

Gets the number of objects in environment.

Returns

The number of objects

6.2.3.10 PhysicalObject * environment::getObject (int id)

Finds an object from the environment.

Parameters

id The id of the object to find

Returns

The object that was looked iup

6.2.3.11 objectIterator environment::getObjectsBegin ()

Gets an iterator to the beginning of the objects.

Returns

The iterator

6.2.3.12 objectIterator environment::getObjectsEnd ()

Gets an iterator to the end of the objects.

Returns

The iterator

6.2.3.13 bool environment::isColliding (Location I, int r)

Checks if the object specified by the given Location and radius is touching a wall or any other objects

1	The Location of the object
r	The radius of the object

Returns

true when the object is touching a wall or any other object

See Also

isTouchingWall, isTouchingObject

6.2.3.14 bool environment::isColliding (int id)

Checks if the object specified by the given id is touching a wall or any other objects

Parameters

id	The id of the object

Returns

true when the object is touching a wall or any other object

See Also

isTouchingWall, isTouchingObject

6.2.3.15 bool environment::isCollidingWithHitable (Location I, int r)

Checks if the object specified by the given Location and radius is touching a wall or any other hitable objects Parameters

1	The Location of the object
r	The radius of the object

Returns

true when the object is touching a wall or any other hitable object

See Also

isTouchingWall, isTouchingObject

6.2.3.16 bool environment::isCollidingWithHitable (int id)

Checks if the object specified by the given id is touching a wall or any other hitable objects

id	The id of the object
----	----------------------

Returns

true when the object is touching a wall or any other hitable object

See Also

isTouchingWall, isTouchingObject

6.2.3.17 bool environment::isTouchingHitableObject (Location I, int r, int id)

Same as isTouchingObject(Location I, int r), but ignores the object with the given id and non-hitable objects.

Parameters

1	The Location of the object
r	The radius of the object
id	The id of the object to ignore

6.2.3.18 bool environment::isTouchingHitableObject (Location l_r int r)

Checks if the object specified by the given Location and radius is touching any other hitable objects

Parameters

1	The Location of the object
r	The radius of the object

Returns

true when the object is touching any other hitable object

6.2.3.19 bool environment::isTouchingHitableObject (int id)

Checks if the object specified by the given id is touching any other hitable objects

Parameters

id	The id of the object

Returns

true when the object is touching any other hitable object

6.2.3.20 bool environment::isTouchingObject (Location I, int r, int id)

Same as isTouchingObject(Location I, int r), but ignores the object with the given id.

Parameters

1	The Location of the object
r	The radius of the object
id	The id of the object to ignore

6.2.3.21 bool environment::isTouchingObject (Location l, int r)

Checks if the object specified by the given Location and radius is touching any other objects

Parameters

1	The Location of the object
r	The radius of the object

Returns

true when the object is touching any other object

6.2.3.22 bool environment::isTouchingObject (int id)

Checks if the object specified by the given id is touching any other objects

Parameters

id	The id of the chiest
Ia	I he id of the object

Returns

true when the object is touching any other object

6.2.3.23 bool environment::isTouchingWall (Location l, int r)

Checks if an object is in the designated window

It returns true whenever the following conditions are met:

```
x position - radius \leq = 0
```

y position - radius \leq = 0

x position + radius >= width

y position + radius >= height

Parameters

ſ	1	The Location of the object
ľ	r	The radius of the object

Returns

true when the object is touching a wall

6.2.3.24 bool environment::isTouchingWall (int id)

Checks if an object is in the designated window

It reads in the width, height, x and y position of the object, and then returns true whenever the following conditions are met:

```
x position - radius <= 0
y position - radius <= 0
x position + radius >= width
y position + radius >= height
```

Parameters

id The id of the object

Returns

true when the object is touching a wall

6.2.3.25 void environment::removeObject (int id)

Removes an object from the environment.

Parameters

id The id of the object to remove

6.3 util Namespace Reference

util namespace, contains helper functions to add and remove robots

Functions

· void reset ()

Removes all objects and resets to the initial state.

· void display ()

Function to render all the objects. This function is called repeatedly from the simulation, and iterates through the objects and calls their respective display functions.

· void advance ()

Function to update the positions of all objects. This function is called repeatedly from the simulation, and iterates through the objects and calls their respective updatePosition functions.

• Color newColor ()

Gets a new, unused color.

• int getNumRobotsTargets ()

Gets the number of robots/target pairs.

• int getNumLights ()

Gets the number of lights.

• int getNumObstacles ()

Gets the number of obstacles.

bool addRobotTarget (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetworkFile)

Adds a robot and paired target to the simulation.

bool addRobot (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robot-SensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetwork-File)

Adds a robot to the simulation.

bool addNeuralNetworkRobotTarget (const NeuralNetwork &network)

Adds a neural network robot and a paired target to the simulation.

• bool addStationaryLightSource ()

Adds a light to the simulation.

bool addMovingLightSource ()

Adds a light to the simulation.

• bool addObstacle ()

Adds a obstacle to the simulation.

bool copy (int id, Location loc)

Copies an object to a new Location.

bool removeRobotTarget ()

Removes a robot from the simulation.

bool removeLightSource ()

Removes a light from the simulation.

bool removeObstacle ()

Removes a obstacle from the simulation.

void removeAllRobotTarget ()

Removes all robots.

void removeAllLightSource ()

Removes all lights.

• void removeAllObstacle ()

Removes all obstacles.

• bool open (std::string filename)

Loads a simulation from a file.

bool save (std::string filename)

Saves the current state to a file.

6.3.1 Detailed Description

util namespace, contains helper functions to add and remove robots

6.3.2 Function Documentation

6.3.2.1 bool util::addMovingLightSource ()

Adds a light to the simulation.

Returns

true if successful

6.3.2.2 bool util::addNeuralNetworkRobotTarget (const NeuralNetwork & network)

Adds a neural network robot and a paired target to the simulation.

Parameters

network	the network to control the robot
---------	----------------------------------

Returns

true if successful

6.3.2.3 bool util::addObstacle ()

Adds a obstacle to the simulation.

Returns

true if successful

6.3.2.4 bool util::addRobot (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetworkFile)

Adds a robot to the simulation.

Parameters

robotType	The type of robot to add. 0 is SimpleRobot, 1 is ComplexRobot, and 2 is NeuralNetworkRobot
lightSensor-	Settings for ComplexRobot light sensor
Connection-	
Pattern	
robotSensor-	Settings for ComplexRobot robot sensor
Connection-	
Pattern	
obstacleSensor-	Settings for ComplexRobot obstacle sensor
Connection-	
Pattern	
targetSensor-	Settings for ComplexRobot target sensor
Connection-	
Pattern	
lightSensorScale	Settings for ComplexRobot light sensor
robotSensor-	Settings for ComplexRobot robot sensor
Scale	
obstacleSensor-	Settings for ComplexRobot obstacle sensor
Scale	
targetSensor-	Settings for ComplexRobot target sensor
Scale	
initalSpeed	Default speed for ComplexRobot
neuralNetwork-	File in which neural network is stored
File	

Returns

true if successful

6.3.2.5 bool util::addRobotTarget (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetworkFile)

Adds a robot and paired target to the simulation.

Parameters

robotType	The type of robot to add. 0 is SimpleRobot, 1 is ComplexRobot, and 2 is NeuralNetworkRobot
lightSensor-	Settings for ComplexRobot light sensor
Connection-	
Pattern	
robotSensor-	Settings for ComplexRobot robot sensor
Connection-	
Pattern	
obstacleSensor-	Settings for ComplexRobot obstacle sensor
Connection-	
Pattern	
targetSensor-	Settings for ComplexRobot target sensor
Connection-	
Pattern	
lightSensorScale	Settings for ComplexRobot light sensor
robotSensor-	Settings for ComplexRobot robot sensor
Scale	
obstacleSensor-	Settings for ComplexRobot obstacle sensor
Scale	
targetSensor-	Settings for ComplexRobot target sensor
Scale	
initalSpeed	Default speed for ComplexRobot
neuralNetwork-	File in which neural network is stored
File	

Returns

true if successful

6.3.2.6 bool util::addStationaryLightSource ()

Adds a light to the simulation.

Returns

true if successful

6.3.2.7 void util::advance ()

Function to update the positions of all objects. This function is called repeatedly from the simulation, and iterates through the objects and calls their respective updatePosition functions.

6.3.2.8 bool util::copy (int id, Location loc)

Copies an object to a new Location.

Parameters

id	the id of the object to copy
loc	the destination

Returns

true if successful

Removes all robots.

```
6.3.2.9 void util::display ( )
Function to render all the objects. This function is called repeatedly from the simulation, and iterates through the
objects and calls their respective display functions.
6.3.2.10 int util::getNumLights ( )
Gets the number of lights.
Returns
      The number of lights
6.3.2.11 int util::getNumObstacles ( )
Gets the number of obstacles.
Returns
      The number of obstacles
6.3.2.12 int util::getNumRobotsTargets ( )
Gets the number of robots/target pairs.
Returns
      The number of robots/target pairs
6.3.2.13 Color util::newColor ( )
Gets a new, unused color.
Returns
      The new color
6.3.2.14 bool util::open ( std::string filename )
Loads a simulation from a file.
6.3.2.15 void util::removeAllLightSource ( )
Removes all lights.
6.3.2.16 void util::removeAllObstacle ( )
Removes all obstacles.
6.3.2.17 void util::removeAllRobotTarget ( )
```

```
6.3.2.18 bool util::removeLightSource ( )
Removes a light from the simulation.
Returns
     true if successful
6.3.2.19 bool util::removeObstacle ( )
Removes a obstacle from the simulation.
Returns
     true if successful
6.3.2.20 bool util::removeRobotTarget ( )
Removes a robot from the simulation.
Returns
      true if successful
6.3.2.21 void util::reset ( )
Removes all objects and resets to the initial state.
6.3.2.22 bool util::save ( std::string filename )
Saves the current state to a file.
```

Chapter 7

Class Documentation

7.1 BaseGfxApp Class Reference

Graphics driver class.

```
#include <BaseGfxApp.h>
```

Inheritance diagram for BaseGfxApp:



Public Member Functions

- BaseGfxApp (int argc, char *argv[], int width, int height, int x, int y, int glutFlags, bool createGLUIWin, int gluiWinX, int gluiWinY)
- virtual ∼BaseGfxApp ()
- void setCaption (const std::string &caption)
- void runMainLoop ()
- virtual void display ()
- virtual void advance ()
- virtual void mouseMoved (int x, int y)
- virtual void mouseDragged (int x, int y)
- virtual void leftMouseDown (int x, int y)
- virtual void leftMouseUp (int x, int y)
- $\bullet \ \ virtual \ void \ \frac{rightMouseDown}{rightMouseDown} \ (int \ x, \ int \ y)$
- virtual void rightMouseUp (int x, int y)
- virtual void middleMouseDown (int x, int y)
- virtual void middleMouseUp (int x, int y)
- virtual void keyboard (unsigned char c, int x, int y)
- virtual void keyboardSpecial (int key, int x, int y)
- virtual void keyboardUp (unsigned char c, int x, int y)
- virtual void keyboardSpecialUp (int key, int x, int y)
- · virtual void reshape (int width, int height)
- virtual void gluiControl (int controlID)
- int width () const
- int height () const
- int handle ()
- GLUI * glui ()

Static Public Member Functions

• static void graphicsBegin (int v, int delay)

Static Protected Member Functions

- static void graphicsTimer (int v)
- static void s_reshape (int width, int height)
- static void s_keyboard (unsigned char c, int x, int y)
- static void s_keyboardspecial (int key, int x, int y)
- static void s keyboardup (unsigned char c, int x, int y)
- static void s_keyboardspecialup (int key, int x, int y)
- static void s_mousemotion (int x, int y)
- static void s_mousebtn (int b, int s, int x, int y)
- static void s_draw ()
- static void s_gluicallback (int controlID)

Protected Attributes

- int m_glutWindowHandle
- GLUI * m_glui
- bool m_drag
- int m_width
- · int m_height

Static Protected Attributes

- static BaseGfxApp * s_currentApp = NULL
- static bool s_glutInitialized = false

7.1.1 Detailed Description

Graphics driver class.

7.1.2 Constructor & Destructor Documentation

- 7.1.2.1 BaseGfxApp::BaseGfxApp (int argc, char * argv[], int width, int height, int x, int y, int glutFlags, bool createGLUIWin, int gluiWinX, int gluiWinY)
- **7.1.2.2** BaseGfxApp::∼BaseGfxApp() [virtual]

7.1.3 Member Function Documentation

7.1.3.1 virtual void BaseGfxApp::advance() [inline], [virtual]

Reimplemented in Simulation.

7.1.3.2 virtual void BaseGfxApp::display() [inline], [virtual]

Reimplemented in Simulation.

```
7.1.3.3 GLUI* BaseGfxApp::glui() [inline]
7.1.3.4 virtual void BaseGfxApp::gluiControl(int controllD) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.5 void BaseGfxApp::graphicsBegin (int v, int delay ) [static]
This function starts the graphics after a delay.
7.1.3.6 void BaseGfxApp::graphicsTimer(int v) [static], [protected]
This function calls itself using glutTimerFunc once per frame.
7.1.3.7 int BaseGfxApp::handle() [inline]
7.1.3.8 int BaseGfxApp::height ( ) const
7.1.3.9 virtual void BaseGfxApp::keyboard (unsigned char c, int x, int y) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.10 virtual void BaseGfxApp::keyboardSpecial (int key, int x, int y ) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.11 virtual void BaseGfxApp::keyboardSpecialUp(int key, int x, int y) [inline], [virtual]
7.1.3.12 virtual void BaseGfxApp::keyboardUp (unsigned char c, int x, int y) [inline], [virtual]
7.1.3.13 virtual void BaseGfxApp::leftMouseDown(int x, int y) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.14 virtual void BaseGfxApp::leftMouseUp(int x, int y) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.15 virtual void BaseGfxApp::middleMouseDown(int x, int y) [inline], [virtual]
Reimplemented in Simulation.
7.1.3.16 virtual void BaseGfxApp::middleMouseUp (int x, int y) [inline], [virtual]
7.1.3.17 virtual void BaseGfxApp::mouseDragged (int x, int y ) [inline], [virtual]
Reimplemented in Simulation.
```

```
7.1.3.18 virtual void BaseGfxApp::mouseMoved(int x, int y) [inline], [virtual]
7.1.3.19 void BaseGfxApp::reshape (int width, int height) [virtual]
7.1.3.20 virtual void BaseGfxApp::rightMouseDown (int x, int y) [inline], [virtual]
7.1.3.21 virtual void BaseGfxApp::rightMouseUp(int x, int y) [inline], [virtual]
7.1.3.22 void BaseGfxApp::runMainLoop()
7.1.3.23 void BaseGfxApp::s_draw( ) [static], [protected]
7.1.3.24 void BaseGfxApp::s_gluicallback(int controllD) [static], [protected]
7.1.3.25 void BaseGfxApp::s_keyboard (unsigned char c, int x, int y) [static], [protected]
7.1.3.26 void BaseGfxApp::s_keyboardspecial (int key, int x, int y) [static], [protected]
7.1.3.27 void BaseGfxApp::s_keyboardspecialup(int key, int x, int y) [static], [protected]
7.1.3.28 void BaseGfxApp::s_keyboardup (unsigned char c, int x, int y) [static], [protected]
7.1.3.29 void BaseGfxApp::s_mousebtn ( int b, int s, int x, int y ) [static], [protected]
7.1.3.30 void BaseGfxApp::s_mousemotion (int x, int y) [static], [protected]
7.1.3.31 void BaseGfxApp::s_reshape(int width, int height) [static], [protected]
7.1.3.32 void BaseGfxApp::setCaption ( const std::string & caption )
7.1.3.33 int BaseGfxApp::width ( ) const
       Member Data Documentation
7.1.4
7.1.4.1 bool BaseGfxApp::m_drag [protected]
7.1.4.2 GLUI* BaseGfxApp::m_glui [protected]
7.1.4.3 int BaseGfxApp::m_glutWindowHandle [protected]
Underlying glut window handle
7.1.4.4 int BaseGfxApp::m_height [protected]
7.1.4.5 int BaseGfxApp::m_width [protected]
7.1.4.6 BaseGfxApp * BaseGfxApp::s_currentApp = NULL [static], [protected]
```

GLUT and GLUI event callbacks are sent to the current window/app. Right now, there is only one window anyway (not counting the GLUI UI window.. in the future could be extended to support more windows. In any case, some structure like this is always needed when using glut with C++, since the glut callbacks must be either global or static functions.

7.2 Color Struct Reference 33

```
7.1.4.7 bool BaseGfxApp::s_glutInitialized = false [static], [protected]
```

Has glutInit been called? (only allowed once per program)

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

- BaseGfxApp.h
- · BaseGfxApp.cpp

7.2 Color Struct Reference

This struct holds the representation of a color.

```
#include <Color.h>
```

Public Member Functions

- Color ()
- Color (float r, float g, float b)
- Color (char c)

Creates a RGB color from its char color name The following is the list of allowed colors:

• Color (int c)

Creates an RGB color from a hex value The color is stored as the last 3 bytes, as 0x<unused (set to 0)><red><green><blue>

Each byte is interpreted as an integer holding an RGB value from 0-255. These are then converted to floats between 0 and 1 by dividing by 255. Note that this requires at least 3 byte integers.

bool isSimilar (Color c1)

Checks if two colors are similar.

- std::ostream & operator<< (std::ostream &out)
- bool operator== (Color c1)

Checks if two colors are equal.

• bool operator!= (Color c1)

Checks if two colors are not equal.

Public Attributes

- · float red
- · float green
- · float blue

7.2.1 Detailed Description

This struct holds the representation of a color.

Each field is a float value that shoult be between 0 and 1. This allows a direct translation from hex colors, with each field being the hex RGB value / 255.

7.2.2 Constructor & Destructor Documentation

```
7.2.2.1 Color::Color( ) [inline]
```

7.2.2.2 Color::Color (float r, float g, float b) [inline]

7.2.2.3 Color::Color (char c)

Creates a RGB color from its char color name The following is the list of allowed colors:

Author

Lucas Kramer

Color char	Color name	RGB floating-point value
'R'	Red	(1, 0, 0)
,O,	Orange	(1, 0.4, 0)
'Y'	Yellow	(1, 1, 0)
'G'	Green	(0, 1, 0)
'B'	Blue	(0, 0, 1)
'V'	Violet	(0.5, 0, 0.5)
'W'	White	(1, 1, 1)
default	Black	(0, 0, 0)

7.2.2.4 Color::Color (int c)

Creates an RGB color from a hex value The color is stored as the last 3 bytes, as 0x-unused (set to 0)><red><green><blue>

Each byte is interpreted as an integer holding an RGB value from 0-255. These are then converted to floats between 0 and 1 by dividing by 255. Note that this requires at least 3 byte integers.

Author

Carl Bahn Lucas Kramer

7.2.3 Member Function Documentation

7.2.3.1 bool Color::isSimilar (Color c1)

Checks if two colors are similar.

Author

Lucas Kramer

Parameters

c1	the color to compare

Returns

true if the colors are similar

7.2.3.2 bool Color::operator!= (Color c1)

Checks if two colors are not equal.

Author

Lucas Kramer

Parameters

c1	the color to check

Returns

true if the colors are not equal

7.2.3.3 std::ostream& Color::operator<< (std::ostream & out) [inline]

7.2.3.4 bool Color::operator== (Color c1)

Checks if two colors are equal.

Author

Lucas Kramer

Parameters

c1	the color to check
----	--------------------

Returns

true if the colors are equal

7.2.4 Member Data Documentation

7.2.4.1 float Color::blue

7.2.4.2 float Color::green

7.2.4.3 float Color::red

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

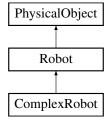
- · Color.h
- · Color.cpp

7.3 ComplexRobot Class Reference

A complex robot with configurable feedback from all sensors.

#include <ComplexRobot.h>

Inheritance diagram for ComplexRobot:



Public Member Functions

- ComplexRobot ()
- ComplexRobot (int radius, Color color, Color lineColor, bool enableLightSensors, bool enableRobotSensors, bool enableObstacleSensors, bool enableTargetSensors, bool lightSensorsCrossed, bool robotSensors-Crossed, bool obstacleSensorsCrossed, bool targetSensorsCrossed, float lightSensorScale, float robot-SensorScale, float obstacleSensorScale, float targetSensorScale, int defaultSpeed, int targetId=-1)
- ComplexRobot (int radius, Location loc, Color color, Color lineColor, bool enableLightSensors, bool enable-RobotSensors, bool enableObstacleSensors, bool enableTargetSensors, bool lightSensorsCrossed, bool robotSensorsCrossed, bool obstacleSensorsCrossed, bool targetSensorsCrossed, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int defaultSpeed, int targetId=-1)
- ComplexRobot ()
- float getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightCobstacleSensorVal, float leftTargetSensorVal, float rightCobstacleSensorVal, float rightTargetSensorVal)
- float getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal)

Public Attributes

- const bool enableLightSensors
- const bool enableRobotSensors
- · const bool enableObstacleSensors
- const bool enableTargetSensors
- const bool lightSensorsCrossed
- const bool robotSensorsCrossed
- · const bool obstacleSensorsCrossed
- const bool targetSensorsCrossed
- const float lightSensorScaleconst float robotSensorScale
- const float obstacleSensorScale
- const float targetSensorScale
- const int defaultSpeed

Additional Inherited Members

7.3.1 Detailed Description

A complex robot with configurable feedback from all sensors.

7.3.2 Constructor & Destructor Documentation

- 7.3.2.1 ComplexRobot::ComplexRobot ()
- 7.3.2.2 ComplexRobot::ComplexRobot (int radius, Color color, Color lineColor, bool enableLightSensors, bool enableRobotSensors, bool enableObstacleSensors, bool enableTargetSensors, bool lightSensorsCrossed, bool robotSensorsCrossed, bool obstacleSensorsCrossed, bool targetSensorsCrossed, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int defaultSpeed, int targetId = -1)
- 7.3.2.3 ComplexRobot::ComplexRobot (int radius, Location loc, Color color, Color lineColor, bool enableLightSensors, bool enableRobotSensors, bool enableObstacleSensors, bool enableTargetSensors, bool lightSensorsCrossed, bool robotSensorsCrossed, bool obstacleSensorsCrossed, bool targetSensorsCrossed, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int defaultSpeed, int targetId = -1)

- 7.3.2.4 ComplexRobot:: ~ComplexRobot ()
- 7.3.3 Member Function Documentation
- 7.3.3.1 float ComplexRobot::getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

7.3.3.2 float ComplexRobot::getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

- 7.3.4 Member Data Documentation
- 7.3.4.1 const int ComplexRobot::defaultSpeed
- 7.3.4.2 const bool ComplexRobot::enableLightSensors
- 7.3.4.3 const bool ComplexRobot::enableObstacleSensors
- 7.3.4.4 const bool ComplexRobot::enableRobotSensors
- 7.3.4.5 const bool ComplexRobot::enableTargetSensors
- 7.3.4.6 const float ComplexRobot::lightSensorScale
- 7.3.4.7 const bool ComplexRobot::lightSensorsCrossed
- 7.3.4.8 const float ComplexRobot::obstacleSensorScale
- 7.3.4.9 const bool ComplexRobot::obstacleSensorsCrossed
- 7.3.4.10 const float ComplexRobot::robotSensorScale
- 7.3.4.11 const bool ComplexRobot::robotSensorsCrossed
- 7.3.4.12 const float ComplexRobot::targetSensorScale
- 7.3.4.13 const bool ComplexRobot::targetSensorsCrossed

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

- · ComplexRobot.h
- ComplexRobot.cpp

7.4 Configuration Class Reference

#include <configuration.h>

Public Member Functions

int getIntConfig (const std::string &name)

Looks up an int.

float getFloatConfig (const std::string &name)

Looks up a float.

bool getBoolConfig (const std::string &name)

Looks up a boolean.

char getCharConfig (const std::string &name)

Looks up a character.

• std::string getStringConfig (const std::string &name)

Looks up a string.

Static Public Member Functions

- static void initConfig (const std::string &filename)
- static void initConfig (int argc, char *argv[], const std::string &defaultFilename)
- static Configuration * get ()
- static void refresh ()

7.4.1 Member Function Documentation

```
7.4.1.1 Configuration * Configuration::get() [static]
```

7.4.1.2 bool Configuration::getBoolConfig (const std::string & name)

Looks up a boolean.

Parameters

table	The table from which to look up the value
name	The name of the value

Returns

The value

7.4.1.3 char Configuration::getCharConfig (const std::string & name)

Looks up a character.

Parameters

table	The table from which to look up the value
name	The name of the value

Returns

The value

7.4.1.4 float Configuration::getFloatConfig (const std::string & name)

Looks up a float.

Parameters

table	The table from which to look up the value
name	The name of the value

Returns

The value

7.4.1.5 int Configuration::getIntConfig (const std::string & name)

Looks up an int.

Parameters

table	The table from which to look up the value
name	The name of the value

Returns

The value

7.4.1.6 string Configuration::getStringConfig (const std::string & name)

Looks up a string.

Parameters

table	The table from which to look up the value
name	The name of the value

Returns

The value

- 7.4.1.7 static void Configuration::initConfig (const std::string & filename) [static]
- 7.4.1.8 static void Configuration::initConfig (int argc, char * argv[], const std::string & defaultFilename) [static]
- 7.4.1.9 void Configuration::refresh () [static]

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

- · configuration.h
- · configuration.cpp

7.5 configValue Struct Reference

This struct holds a configuration value of any legal type.

#include <configuration.h>

Public Attributes

```
    std::string type
    union {
        int intVal
        float floatVal
        char charVal
        bool boolVal
        const char * stringVal
    };
```

7.5.1 Detailed Description

This struct holds a configuration value of any legal type.

Note that stringVal is stored as a const char * in the union due to restrictions in C++ not allowing complex types in unions. The string is converted to a char * after it is parsed, and is converted back in getStringVal.

7.5.2 Member Data Documentation

```
7.5.2.1 union { ... }
```

An anonymous union of all possible types that can be stored.

```
7.5.2.2 bool configValue::boolVal
```

7.5.2.3 char configValue::charVal

7.5.2.4 float configValue::floatVal

7.5.2.5 int configValue::intVal

7.5.2.6 const char* configValue::stringVal

7.5.2.7 std::string configValue::type

The name of the type that is stored ("int", "float", "char", "bool", or "string") or an error code from parsing ("invalid type name" or "invalid syntax").

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

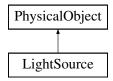
· configuration.h

7.6 LightSource Class Reference

LightSource for the robot to seek.

```
#include <LightSource.h>
```

Inheritance diagram for LightSource:



Public Member Functions

- LightSource (int radius, Color color)
- LightSource (int maxRadius, int minRadius, Color color)
- LightSource (int radius, Location loc, Color color)
- bool handleCollision (int otherId, bool wasHit)
- ∼LightSource ()
- void display ()

Additional Inherited Members

7.6.1 Detailed Description

LightSource for the robot to seek.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 LightSource::LightSource (int radius, Color color)

This constructor requires two parameters: radius and color)

Parameters

radius	the radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color

Initialization

LightSource is initialized with the following values:

Orientation: random

Speed: 0

Position: (loc.x, loc.y)

7.6.2.2 LightSource::LightSource (int maxRadius, int minRadius, Color color)

This constructor creates a light with a random radius

Parameters

minRadius	the minimum possible radius in pixels
maxRadius	the maximum possible radius in pixels
color	a struct representing color

Initialization

LightSource is initialized with the following values:

Orientation: random

Speed: 0

Position: (loc.x, loc.y)

7.6.2.3 LightSource::LightSource (int radius, Location loc, Color color)

This constructor requires four parameters: radius, color, start Location and ConfigTable value)

Parameters

radius	the radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color
loc	a struct of the x,y start Location in pixels
config	a configuration table

Initialization

LightSource is initialized with the following values:

Orientation: 0

Speed: 0

Position: (loc.x, loc.y)

7.6.2.4 LightSource::~LightSource()

This is the class destructor.

7.6.3 Member Function Documentation

7.6.3.1 void LightSource::display() [virtual]

This function displays the LightSource in the graphics.

Reimplemented from PhysicalObject.

7.6.3.2 bool LightSource::handleCollision (int otherId, bool wasHit) [virtual]

Author

Lucas Kramer This function gives LightSource an appropriate reaction when a collision occurs.

Parameters

otherId	id of the object that it collides with
wasHit	tells if the object was hit previously

Implements PhysicalObject.

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

- · LightSource.h
- · LightSource.cpp

7.7 Location Struct Reference

an xy position on the screen, approximately in pixels x is horizontal from the left and y is vertical from the bottom #include <Location.h>

Public Member Functions

- · Location ()
- Location (float x, float y)
- bool operator== (Location other)

Public Attributes

- float x
- float y

7.7.1 Detailed Description

an xy position on the screen, approximately in pixels x is horizontal from the left and y is vertical from the bottom

7.7.2 Constructor & Destructor Documentation

```
7.7.2.1 Location::Location() [inline]
```

7.7.2.2 Location::Location (float x, float y) [inline]

7.7.3 Member Function Documentation

```
7.7.3.1 bool Location::operator== ( Location other ) [inline]
```

7.7.4 Member Data Documentation

7.7.4.1 float Location::x

7.7.4.2 float Location::y

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

· Location.h

7.8 NeuralNetwork Class Reference

A representation of a neural network.

```
#include <NeuralNetwork.h>
```

Public Member Functions

- NeuralNetwork (const NeuralNetwork &other)
- NeuralNetwork (const std::string &filename)
- std::vector< float > compute (const std::vector< float > &inputs)

- · NeuralNetwork mutate (int numChanged, float amount)
- NeuralNetwork combine (const NeuralNetwork &other)
- void write (const std::string &filename)

Static Public Member Functions

static NeuralNetwork load (const std::string &filename)

7.8.1 Detailed Description

A representation of a neural network.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 NeuralNetwork::NeuralNetwork (const NeuralNetwork & other)

The copy constructor

Parameters

other	the network to copy
-------	---------------------

7.8.2.2 NeuralNetwork::NeuralNetwork (const std::string & filename)

Constructs a network from a file description

Parameters

filename	the description filename

7.8.3 Member Function Documentation

7.8.3.1 NeuralNetwork NeuralNetwork::combine (const NeuralNetwork & other)

Combines the network with another network by constructing a new network with half the connection strengths from each. Throws an exception if the networks have incompatible structures

Returns

the new network

7.8.3.2 vector < float > NeuralNetwork::compute (const std::vector < float > & inputs)

Computes the output values of a network for a vector of inputs. Throws an exception if an incorrect number of inputs is provided

Parameters

inputs	a vector containing the inputs

Returns

a vector containing the outputs

7.8.3.3 NeuralNetwork NeuralNetwork::load (const std::string & filename) [static]

Loads a network from a file

Parameters

filename	the filename to load
----------	----------------------

Returns

the new network

7.8.3.4 NeuralNetwork NeuralNetwork::mutate (int numChanged, float amount)

Mutates the network by changing numChanged connection strengths by a random number between -amount and amount

Parameters

numChanged	the number of connections to mutate
amount	the maximum amount to change the connections

Returns

the new network

7.8.3.5 void NeuralNetwork::write (const std::string & filename)

Writes the network to a file

Parameters

filename	the file to write

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

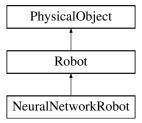
- · NeuralNetwork.h
- NeuralNetwork.cpp

7.9 NeuralNetworkRobot Class Reference

A robot controlled by a neural network with inputs from the sensors.

#include <NeuralNetworkRobot.h>

Inheritance diagram for NeuralNetworkRobot:



Public Member Functions

 NeuralNetworkRobot (int radius, Color color, Color lineColor, NeuralNetwork network, int targetId=-1, std-::string filename=GET_STRING("DEFAULT_NEURAL_NETWORK_FILE"))

- NeuralNetworkRobot (int radius, Location loc, Color color, Color lineColor, NeuralNetwork network, int targetld=-1, std::string filename=GET_STRING("DEFAULT_NEURAL_NETWORK_FILE"))
- ∼NeuralNetworkRobot ()
- float getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal)
- float getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal)

Public Attributes

• const std::string filename

Additional Inherited Members

7.9.1 Detailed Description

A robot controlled by a neural network with inputs from the sensors.

- 7.9.2 Constructor & Destructor Documentation
- 7.9.2.1 NeuralNetworkRobot::NeuralNetworkRobot (int radius, Color color, Color lineColor, NeuralNetwork network, int targetId = -1, std::string filename = GET_STRING ("DEFAULT_NEURAL_NETWORK_FILE"))
- 7.9.2.2 NeuralNetworkRobot::NeuralNetworkRobot (int radius, Location loc, Color color, Color lineColor, NeuralNetwork network, int targetId = -1, std::string filename = GET_STRING ("DEFAULT_NEURAL_NETWORK_FILE"))
- 7.9.2.3 NeuralNetworkRobot::~NeuralNetworkRobot ()
- 7.9.3 Member Function Documentation
- 7.9.3.1 float NeuralNetworkRobot::getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftTargetSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

7.9.3.2 float NeuralNetworkRobot::getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

7.9.4 Member Data Documentation

7.9.4.1 const std::string NeuralNetworkRobot::filename

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

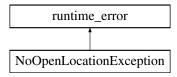
- · NeuralNetworkRobot.h
- NeuralNetworkRobot.cpp

7.10 NoOpenLocationException Class Reference

An exception to be thrown when an open Location cannot be found.

```
#include <environment.h>
```

Inheritance diagram for NoOpenLocationException:



Public Member Functions

NoOpenLocationException ()

7.10.1 Detailed Description

An exception to be thrown when an open Location cannot be found.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 NoOpenLocationException::NoOpenLocationException() [inline]

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

· environment.h

7.11 ObjectIterator Class Reference

ObjectIterator class, a simple iterator for the object in environment.

```
#include <environment.h>
```

Public Member Functions

• ObjectIterator ()

The default constructor, the iterated objects are allways the iterObjects vector in environment.

• ObjectIterator (const ObjectIterator &other)

The copy constructor, the iterated objects are allways the iterObjects vector in environment.

∼ObjectIterator ()

The destructor, does nothing.

- void operator++ ()
- void operator++ (int)
- PhysicalObject * operator* ()
- bool operator> (const ObjectIterator &other)
- bool operator< (const ObjectIterator &other)
- bool operator== (const ObjectIterator &other)
- bool operator!= (const ObjectIterator &other)

Friends

- environment::objectIterator environment::getObjectsBegin ()
- environment::objectIterator environment::getObjectsEnd ()

7.11.1 Detailed Description

ObjectIterator class, a simple iterator for the object in environment.

7.11.2 Constructor & Destructor Documentation

```
7.11.2.1 ObjectIterator::ObjectIterator()
```

The default constructor, the iterated objects are allways the iterObjects vector in environment.

```
7.11.2.2 ObjectIterator::ObjectIterator ( const ObjectIterator & other )
```

The copy constructor, the iterated objects are allways the iterObjects vector in environment.

Parameters

other The ObjectIterator to copy

```
7.11.2.3 ObjectIterator::~ObjectIterator()
```

The destructor, does nothing.

7.11.3 Member Function Documentation

```
7.11.3.1 bool ObjectIterator::operator!= ( const ObjectIterator & other )
```

```
7.11.3.2 PhysicalObject * ObjectIterator::operator* ( )
```

- 7.11.3.3 void ObjectIterator::operator++ ()
- 7.11.3.4 void ObjectIterator::operator++ (int)
- 7.11.3.5 bool ObjectIterator::operator < (const ObjectIterator & other)
- 7.11.3.6 bool ObjectIterator::operator== (const ObjectIterator & other)
- 7.11.3.7 bool ObjectIterator::operator> (const ObjectIterator & other)

7.11.4 Friends And Related Function Documentation

- 7.11.4.1 environment::objectIterator environment::getObjectsBegin() [friend]
- 7.11.4.2 environment::objectIterator environment::getObjectsEnd() [friend]

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

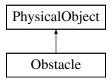
- · environment.h
- environment.cpp

7.12 Obstacle Class Reference

Obstacle for the robot to hit/avoid.

#include <Obstacle.h>

Inheritance diagram for Obstacle:



Public Member Functions

- Obstacle (int radius, Color color)
- Obstacle (int maxRadius, int minRadius, Color color)
- Obstacle (int radius, Location loc, Color color)
- bool handleCollision (int otherId, bool wasHit)
- ∼Obstacle ()

Additional Inherited Members

7.12.1 Detailed Description

Obstacle for the robot to hit/avoid.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Obstacle::Obstacle (int radius, Color color)

Parameters

radius	the radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color
config	a configuration table

Initialization

Obstacle is initialized with the following values:

Radius: DEFAULT_RADIUS as specified in ../config/default

Orientation: random

Speed: 0

Position: (loc.x, loc.y)

7.12.2.2 Obstacle::Obstacle (int maxRadius, int minRadius, Color color)

Parameters

minRadius	the minimum radius in pixels
maxRadius	the maximum radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color

Initialization

Obstacle is initialized with the following values:

Orientation: random

Speed: 0

Position: (loc.x, loc.y)

7.12.2.3 Obstacle::Obstacle (int radius, Location loc, Color color)

Parameters

radius	the radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color
loc	a struct of the x,y start Location in pixels

Initialization

Obstacle is initialized with the following values:

Orientation: 0

Speed: 0

Position: (loc.x, loc.y)

7.12.2.4 Obstacle::~Obstacle()

This is the class destructor.

7.12.3 Member Function Documentation

7.12.3.1 bool Obstacle::handleCollision (int otherId, bool wasHit) [virtual]

Author

Lucas Kramer This function gives Obstacle an appropriate reaction when a collision occurs. Does nothing for now

Parameters

otherId	id of the object that it collides with
wasHit	tells if the object was hit previously

Implements PhysicalObject.

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

- · Obstacle.h
- Obstacle.cpp

7.13 OptimizeSimulation Class Reference

OptimizeSimulation class, sets up environments and robots.

```
#include <OptimizeSimulation.h>
```

Public Member Functions

• OptimizeSimulation (int argc, char *argv[])

The constructor for the Simulation class.

- virtual ∼OptimizeSimulation ()
- void runMainLoop ()
- int getPerformance (const NeuralNetwork &network)
- int getPerformanceRandomRepeated (const NeuralNetwork &network)
- int getPerformanceMaze (const NeuralNetwork &network)
- int getPerformanceObstacles (const NeuralNetwork &network)

7.13.1 Detailed Description

OptimizeSimulation class, sets up environments and robots.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 OptimizeSimulation::OptimizeSimulation (int argc, char * argv[])

The constructor for the Simulation class.

Parameters

argc	The number of command-line arguments
argv	The command-line arguments

- 7.13.2.2 OptimizeSimulation::~OptimizeSimulation() [virtual]
- 7.13.3 Member Function Documentation
- 7.13.3.1 int OptimizeSimulation::getPerformance (const NeuralNetwork & network)
- 7.13.3.2 int OptimizeSimulation::getPerformanceMaze (const NeuralNetwork & network)
- 7.13.3.3 int OptimizeSimulation::getPerformanceObstacles (const NeuralNetwork & network)
- 7.13.3.4 int OptimizeSimulation::getPerformanceRandomRepeated (const NeuralNetwork & network)
- 7.13.3.5 void OptimizeSimulation::runMainLoop ()

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

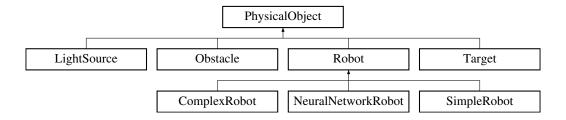
- · OptimizeSimulation.h
- · OptimizeSimulation.cpp

7.14 PhysicalObject Class Reference

This is a common superclass for all objects.

#include <PhysicalObject.h>

Inheritance diagram for PhysicalObject:



Public Member Functions

- PhysicalObject (ObjectType objectType, int radius=GET_INT("DEFAULT_RADIUS"), Color color=Color('?'), bool isHitable=true)
- PhysicalObject (ObjectType objectType, int maxRadius, int minRadius, Color color=Color('?'), bool is-Hitable=true)
- PhysicalObject (ObjectType objectType, int radius, Location loc, Color color=Color('?'), bool isHitable=true)
- virtual ∼PhysicalObject ()
- int getId () const
- void setPosition (float x, float y)
- void forceSetPosition (float x, float y)
- void setLocation (Location loc)
- void reorient (int angle, float distance)
- bool translate (float distance)
- void forceTranslate (float distance)
- void setOrientation (int degrees)
- void setSpeed (int pps)
- void setRadius (int radius)
- void setColor (Color color)
- float getXPosition () const
- float getYPosition () const
- · Location getLocation () const
- int getOrientation () const
- int getSpeed () const
- int getRadius () const
- Color getColor () const
- void pointTo (const PhysicalObject &other)
- void rotate (int degrees)
- bool hasEqualPosition (const PhysicalObject &other) const
- bool updatePosition ()
- virtual void update ()
- virtual void updateMembers ()
- virtual void display ()
- virtual bool handleCollision (int otherId, bool wasHit)=0

Public Attributes

- const ObjectType objectType
- · const bool isHitable

Static Protected Member Functions

static Location findOpenLocation (int radius)

7.14.1 Detailed Description

This is a common superclass for all objects.

7.14.2 Constructor & Destructor Documentation

```
7.14.2.1 PhysicalObject::PhysicalObject ( ObjectType objectType, int radius = GET_INT ("DEFAULT_RADIUS"), Color color = Color ('?'), bool isHitable = true )
```

7.14.2.2 PhysicalObject::PhysicalObject (ObjectType objectType, int maxRadius, int minRadius, Color color = Color ('?'), bool isHitable = true)

7.14.2.3 PhysicalObject::PhysicalObject (ObjectType objectType, int radius, Location loc, Color color = Color ('?'), bool isHitable = true)

7.14.2.4 PhysicalObject::~PhysicalObject() [virtual]

This is the class destructor, it is implemented by all subclasses.

7.14.3 Member Function Documentation

```
7.14.3.1 void PhysicalObject::display() [virtual]
```

This is implemented by each class, normally just calls displayAsCircle.

See Also

displayAsCircle()

Reimplemented in Robot, and LightSource.

7.14.3.2 Location PhysicalObject::findOpenLocation (int radius) [static], [protected]

This function finds an open Location to place the object

Parameters

radius	The radius of the object

7.14.3.3 void PhysicalObject::forceSetPosition (float x, float y)

Sets the position for the object. It does not throw an exception on invalid input.

Parameters

X	position in pixels
у	position in pixels

7.14.3.4 void PhysicalObject::forceTranslate (float distance)

Translates without calling handleCollision. This may cause objects to overlap

Parameters

Returns

speed in pixels per second

distance Distance to move in pixels 7.14.3.5 Color PhysicalObject::getColor () const Returns the color of the object Returns color 7.14.3.6 int PhysicalObject::getId () const Getter function for the object's id. Returns The object's id 7.14.3.7 Location PhysicalObject::getLocation () const Returns the x and y Location of the object in pixels in a struct See Also Location 7.14.3.8 int PhysicalObject::getOrientation () const Returns the orientation of the object in degrees Returns orientation in degrees 7.14.3.9 int PhysicalObject::getRadius () const Returns the radius of the object Returns radius 7.14.3.10 int PhysicalObject::getSpeed () const Returns the speed of the object

7.14.3.11 float PhysicalObject::getXPosition () const

Returns the x position in pixels of the object

Returns

x position, horizontal from left

7.14.3.12 float PhysicalObject::getYPosition () const

Returns the y position in pixels of the object

Returns

y position, vertical from bottom

7.14.3.13 virtual bool PhysicalObject::handleCollision (int otherId, bool wasHit) [pure virtual]

This is implemented by each class. It controls the objects behavior on collision during translate, normally just calls reorient.

Parameters

otherId	The id of the object it hit, or -1 for a wall
wasHit	true if the object hit somthing, false if hit by somthing (see translate)

Returns

true if objects were added or removed

See Also

reorient()

Implemented in Robot, LightSource, Obstacle, and Target.

7.14.3.14 bool PhysicalObject::hasEqualPosition (const PhysicalObject & other) const

This member function checks when two objects are at the same Location, it has one parameter

Parameters

other	The object to compare

Returns

A boolean value that is true when objects are overlapping

7.14.3.15 void PhysicalObject::pointTo (const PhysicalObject & other)

Calculates the direction of the object is facing towards the target object based on position of the two objects using the math function atan2

Parameters

other	A PhysicalObject class pointer

7.14.3.16 void PhysicalObject::reorient (int angle, float distance)

This function can be called by subclasses when handleing a collision. This causes the robot to rotate by the specified amount, and then translate if that is enabled in the configuration. Since translating could cause another collision, this could call back to reorient, causing infinite recusion in some cases. To prevent this, the function ommits the translate if it is more than MAX_REORIENT_RETRIES deep in recursive calls, set in the configuration

Parameters

angle	Angle to rotate in degrees
distance	Distance to move in pixels

7.14.3.17 void PhysicalObject::rotate (int degrees)

This function increments the object's orientation

Parameters

degrees	Angle to be incremented clockwise.
---------	------------------------------------

Exceptions

Input	must be within (-360,360)

7.14.3.18 void PhysicalObject::setColor (Color color)

Set a new color for the object.

Parameters

color	•
-------	---

See Also

Color

7.14.3.19 void PhysicalObject::setLocation (Location loc)

Sets the position for the object for the object

Parameters

loc	A struct of the x,y Location in pixels

See Also

setPosition

7.14.3.20 void PhysicalObject::setOrientation (int degrees)

Sets the absolute orientation for an object

Parameters

degrees	New angle for the object in degrees clockwise of North.
---------	---

7.14.3.21 void PhysicalObject::setPosition (float x, float y)

Sets the position for the object

Parameters

X	x position in pixels
У	y position in pixels

7.14.3.22 void PhysicalObject::setRadius (int radius)

Set the radius of the object

Parameters

radius	the new radius in pixels
--------	--------------------------

7.14.3.23 void PhysicalObject::setSpeed (int pps)

Sets the speed of an object

Parameters

pps	The new speed in pixels per second
-----	------------------------------------

7.14.3.24 bool PhysicalObject::translate (float distance)

Translates in the current direction, and calls handleCollision where needed

Parameters

distance

Returns

true if objects were added or removed when any collisions were handled. This is to allow the position update function to break out of the loop when objects are removed, to prevent trying to update objects that no longer exist

7.14.3.25 void PhysicalObject::update() [virtual]

This is implemented by each class, and performs random updates, e.g. point toward target Reimplemented in Robot.

7.14.3.26 void PhysicalObject::updateMembers() [virtual]

This is implemented by each class, and perfoms updates such as setting the position of sub-objects Reimplemented in Robot.

7.14.3.27 bool PhysicalObject::updatePosition ()

This member function uses the speed and time to update the Location

Returns

true if objects were added or removed

7.14.4 Member Data Documentation

7.14.4.1 const bool PhysicalObject::isHitable

7.14.4.2 const ObjectType PhysicalObject::objectType

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

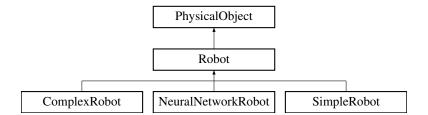
- · PhysicalObject.h
- · PhysicalObject.cpp

7.15 Robot Class Reference

Robot that moves around the window and bumps into obstacles.

```
#include <Robot.h>
```

Inheritance diagram for Robot:



Public Member Functions

- Robot (RobotType robotType, int radius, Color color, Color lineColor, int targetId=-1)
- Robot (RobotType robotType, int radius, Location loc, Color color, Color lineColor, int targetId=-1)
- ∼Robot ()
- int getTarget () const
- void setTarget (int id)
- · bool handleCollision (int otherId, bool wasHit)
- void update ()
- void updateMembers ()
- void display ()
- void setLineColor (Color lineColor)
- · Color getLineColor ()

Public Attributes

const RobotType robotType

Protected Member Functions

- virtual float getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal)=0
- virtual float getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal)=0

Additional Inherited Members

7.15.1 Detailed Description

Robot that moves around the window and bumps into obstacles.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Robot::Robot (RobotType robotType, int radius, Color color, Color lineColor, int targetId = -1)

Parameters

	robotType	the robot type
	radius	the radius in pixels
	color	color of the robot itself, a struct of float value 0 to 1 representing a hexedecimal color
Ì	lineColor	color of the line, a struct of float value 0 to 1 representing a hexedecimal color

Initialization

Robot is initialized with the following values:

Orientation: 0

Speed: 0

Position: (loc.x, loc.y)

7.15.2.2 Robot::Robot (RobotType robotType, int radius, Location loc, Color color, Color lineColor, int targetId = -1)

Parameters

robotType	the robot type
radius	the radius in pixels
color	a struct of float value 0 to 1 representing a hexedecimal color
linecolor	color of the line, a struct of float value 0 to 1 representing a hexedecimal color
loc	start Location of the robot

Initialization

Robot is initialized with the following values:

Orientation: random

Speed: 0

Position: (loc.x, loc.y)

7.15.2.3 Robot::∼Robot ()

This is the class destructor.

7.15.3 Member Function Documentation

7.15.3.1 void Robot::display() [virtual]

Author

Lucas Kramer This function displays the Robot in the graphics

Reimplemented from PhysicalObject.

7.15.3.2 virtual float Robot::getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal) [protected], [pure virtual]

Implemented in ComplexRobot, NeuralNetworkRobot, and SimpleRobot.

- 7.15.3.3 Color Robot::getLineColor ()
- 7.15.3.4 virtual float Robot::getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal) [protected], [pure virtual]

Implemented in ComplexRobot, NeuralNetworkRobot, and SimpleRobot.

7.15.3.5 int Robot::getTarget () const

Author

Lucas Kramer This function gets the id of the target for the robot to seek.

Parameters

id	the id of the target

7.15.3.6 bool Robot::handleCollision (int otherId, bool wasHit) [virtual]

Author

Lucas Kramer This function gives Robot an appropriate reaction when a collision occurs

Parameters

otherId	id of the object that it collides with
wasHit	tells if the object was hit previously

Implements PhysicalObject.

7.15.3.7 void Robot::setLineColor (Color lineColor)

Author

Lucas Kramer This function sets a new color for the robot's direction line.

Parameters

lineColor | color of the line, a struct of float value 0 to 1 representing a hexedecimal color

7.15.3.8 void Robot::setTarget (int id)

Author

Lucas Kramer This function sets the id of the target for the robot to seek.

Returns

the target id

7.15.3.9 void Robot::update() [virtual]

Author

Lucas Kramer This function updates the Robot's wheel speeds from the sensors

Reimplemented from PhysicalObject.

7.15.3.10 void Robot::updateMembers() [virtual]

Author

Lucas Kramer This function updates the position of the robot's sub-objects

Reimplemented from PhysicalObject.

7.15.4 Member Data Documentation

7.15.4.1 const RobotType Robot::robotType

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

- Robot.h
- Robot.cpp

7.16 Sensor Class Reference

```
#include <Sensor.h>
```

Public Member Functions

- Sensor (Location loc, int orientation, ObjectType typeDetected)
- Sensor (Location loc, int orientation, int viewAngle, ObjectType typeDetected)
- void setPosition (float x, float y)
- void setPosition (Location loc)
- · Location getPosition ()
- float getXPosition ()
- float getYPosition ()

- void setOrientation (int orientation)
- int getOrientation ()
- void setViewAngle (int degrees)
- int getViewAngle ()
- void setTypeDetected (ObjectType type)
- ObjectType getTypeDetected ()
- void updatePosition (Location robotLoc, int robotAngle)
- void display ()
- float sense ()

7.16.1 Constructor & Destructor Documentation

7.16.1.1 Sensor::Sensor (Location loc, int orientation, ObjectType typeDetected)

Sensor constructor with default angle

Parameters

loc	Location of the sensor in x and y
orientation	orientation of the sensor
typeDetected	type that sensor will detect

7.16.1.2 Sensor::Sensor (Location loc, int orientation, int viewAngle, ObjectType typeDetected)

Sensor constructor

Parameters

loc	Location of the sensor in x and y
orientation	orientation of the sensor
viewAngle	view angle of the sensor in degrees
typeDetected	type that sensor will detect

7.16.2 Member Function Documentation

7.16.2.1 void Sensor::display ()

Displays the sensor

Author

Carl

7.16.2.2 int Sensor::getOrientation ()

Returns the orientation of the sensor

Returns

orientation in degrees

```
7.16.2.3 Location Sensor::getPosition ( )
Returns a Location struct of x and y position in pixels of the sensor
Returns
      A Location struct of x and y position
7.16.2.4 ObjectType Sensor::getTypeDetected ( )
Returns the type that sensor detects
Author
      Himawan
Returns
     type the sensor detects
7.16.2.5 int Sensor::getViewAngle ( )
Returns the view angle of the sensor
Returns
     view angle in degrees
7.16.2.6 float Sensor::getXPosition ( )
Returns the x position in pixels of the sensor
Returns
     x position
7.16.2.7 float Sensor::getYPosition ( )
Returns the y position in pixels of the sensor
Returns
     y position
7.16.2.8 float Sensor::sense ( )
Returns some strength by sensing the environment
Author
      Carl & Himawan
Returns
      the strength of sensor reading of [0..1]
```

7.16.2.9 void Sensor::setOrientation (int orientation)

Sets the orientation for the sensor

Parameters

orientation	The orientation of the object in degrees
-------------	--

7.16.2.10 void Sensor::setPosition (float x, float y)

Sets the position for the sensor

Parameters

X	x position in pixels
у	y position in pixels

7.16.2.11 void Sensor::setPosition (Location loc)

Sets the position for the sensor

Parameters

loc	A struct of the x and y Location in pixels

7.16.2.12 void Sensor::setTypeDetected (ObjectType type)

Sets the object type that the sensor detects

Author

Himawan

Parameters

type	object type to detect

7.16.2.13 void Sensor::setViewAngle (int degrees)

Sets the view angle of the sensor

Parameters

degrees	The view angle of the sensor in degrees

7.16.2.14 void Sensor::updatePosition (Location robotLoc, int robotAngle)

Called to update the position of the sensor match the updated position of the robot

Author

Carl

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

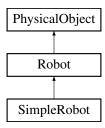
- · Sensor.h
- · Sensor.cpp

7.17 SimpleRobot Class Reference

A simple robot with uncrossed feedback.

#include <SimpleRobot.h>

Inheritance diagram for SimpleRobot:



Public Member Functions

- SimpleRobot (int radius, Color color, Color lineColor, int targetId=-1)
- SimpleRobot (int radius, Location loc, Color color, Color lineColor, int targetId=-1)
- ∼SimpleRobot ()
- float getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightCbstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal, float rightTargetSensorVal)
- float getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightCobstacleSensorVal, float leftTargetSensorVal, float rightTargetSensorVal, float rightTargetSensorVal)

Additional Inherited Members

7.17.1 Detailed Description

A simple robot with uncrossed feedback.

7.17.2 Constructor & Destructor Documentation

- 7.17.2.1 SimpleRobot::SimpleRobot (int radius, Color color, Color lineColor, int targetId = -1)
- 7.17.2.2 SimpleRobot::SimpleRobot (int radius, Location loc, Color color, Color lineColor, int targetId = -1)
- 7.17.2.3 SimpleRobot::~SimpleRobot()

7.17.3 Member Function Documentation

7.17.3.1 float SimpleRobot::getLeftSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

7.17.3.2 float SimpleRobot::getRightSpeed (float leftLightSensorVal, float rightLightSensorVal, float leftRobotSensorVal, float rightRobotSensorVal, float leftObstacleSensorVal, float rightObstacleSensorVal, float rightTargetSensorVal) [virtual]

Implements Robot.

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

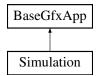
- · SimpleRobot.h
- SimpleRobot.cpp

7.18 Simulation Class Reference

Simulation class, sets up the GUI and the drawing environment.

```
#include <Simulation.h>
```

Inheritance diagram for Simulation:



Public Types

```
    enum UlControlType {
    UI_QUIT = 0, UI_START = 1, UI_PAUSE = 2, UI_RESUME = 3,
    UI_RESET = 4, UI_ADD_RT = 5, UI_REMOVE_RT = 6, UI_REMOVE_ALL_RT = 7,
    UI_ADD_O = 8, UI_REMOVE_O = 9, UI_REMOVE_ALL_O = 10, UI_REFRESH_SETTINGS_O = 11 }
```

Public Member Functions

• Simulation (int argc, char *argv[])

The constructor for the Simulation class.

- virtual ∼Simulation ()
- void initObjects ()

Adds objects according to configuration.

void tryAddRobotTarget ()

Adds a robot and target to the simulation, or raises an error on failure.

void tryAddRobot ()

Adds a robot to the simulation, or raises an error on failure.

• void tryAddStationaryLightSource ()

Adds a stationary light to the simulation, or raises an error on failure.

void tryAddMovingLightSource ()

Adds a moving light to the simulation, or raises an error on failure.

• void tryAddObstacle ()

Adds a obstacle to the simulation, or raises an error on failure.

void tryRemoveRobotTarget ()

Removes a robot and target from the simulation, or raises an error on failure.

void tryRemoveLightSource ()

Removes a light from the simulation, or raises an error on failure.

void tryRemoveObstacle ()

Removes a obstacle from the simulation, or raises an error on failure.

· void tryOpen ()

Loads a simulation, or raises an error on failure.

void trySave ()

Saves a simulation, or raises an error on failure.

· void display ()

Call-back function to render all the objects. This function is called repeatedly from glut, and iterates through the objects and calls their respective display functions.

· void advance ()

Call-back function to update the positions of all objects. This function is called repeatedly from glut, and iterates through the objects and calls their respective updatePosition functions.

• void gluiControl (int controlID)

Call-back function for user input.

• void leftMouseDown (int x, int y)

Call-back function for mouse click.

void leftMouseUp (int x, int y)

Call-back function for mouse click.

void middleMouseDown (int x, int y)

Call-back function for mouse click.

void mouseDragged (int x, int y)

Call-back function for mouse drag.

void keyboard (unsigned char key, int x, int y)

Call-back function keyboard press.

void keyboardSpecial (int key, int x, int y)

Call-back function for special keyboard press.

- · void start (int)
- · void pause (int)
- · void resume (int)
- · void reinit (int)

Static Public Member Functions

- static void s_start (int)
- static void s_pause (int)
- static void s_resume (int)
- static void s_reinit (int)
- static void s_addRobotTarget (int)
- static void s_addRobot (int)
- static void s_addStationaryLightSource (int)
- static void s addMovingLightSource (int)
- static void s_addObstacle (int)
- static void s_removeRobotTarget (int)
- static void s_removeLightSource (int)
- static void s removeObstacle (int)
- static void s removeAllRobotTarget (int)
- static void s_removeAllLightSource (int)
- static void s_removeAllObstacle (int)
- static void s_refreshConfiguration (int)
- static void s_open (int)
- static void s_save (int)
- static void s_neuralNetworkFileChanged (int)

Additional Inherited Members

7.18.1 Detailed Description

Simulation class, sets up the GUI and the drawing environment.

7.18.2 Member Enumeration Documentation

7.18.2.1 enum Simulation::UIControlType

Enumerator

```
UI_QUIT
UI_START
UI_PAUSE
UI_RESUME
UI_RESET
UI_ADD_RT
UI_REMOVE_RT
UI_REMOVE_ALL_RT
UI_ADD_O
UI_REMOVE_O
UI_REMOVE_ALL_O
UI_REFRESH_SETTINGS_O
```

7.18.3 Constructor & Destructor Documentation

7.18.3.1 Simulation::Simulation (int argc, char * argv[])

The constructor for the Simulation class.

Parameters

	argc	The number of command-line arguments
İ	argv	The command-line arguments

```
7.18.3.2 Simulation::~Simulation() [virtual]
```

7.18.4 Member Function Documentation

```
7.18.4.1 void Simulation::advance( ) [virtual]
```

Call-back function to update the positions of all objects. This function is called repeatedly from glut, and iterates through the objects and calls their respective updatePosition functions.

Reimplemented from BaseGfxApp.

```
7.18.4.2 void Simulation::display ( ) [virtual]
```

Call-back function to render all the objects. This function is called repeatedly from glut, and iterates through the objects and calls their respective display functions.

Reimplemented from BaseGfxApp.

```
7.18.4.3 void Simulation::gluiControl (int controlID) [virtual]
Call-back function for user input.
Reimplemented from BaseGfxApp.
7.18.4.4 void Simulation::initObjects ( )
Adds objects according to configuration.
7.18.4.5 void Simulation::keyboard (unsigned char key, int x, int y) [virtual]
Call-back function keyboard press.
Reimplemented from BaseGfxApp.
7.18.4.6 void Simulation::keyboardSpecial (int key, int x, int y) [virtual]
Call-back function for special keyboard press.
Reimplemented from BaseGfxApp.
7.18.4.7 void Simulation::leftMouseDown (int x, int y ) [virtual]
Call-back function for mouse click.
Reimplemented from BaseGfxApp.
7.18.4.8 void Simulation::leftMouseUp (int x, int y) [virtual]
Call-back function for mouse click.
Reimplemented from BaseGfxApp.
7.18.4.9 void Simulation::middleMouseDown (int x, int y ) [virtual]
Call-back function for mouse click.
Reimplemented from BaseGfxApp.
7.18.4.10 void Simulation::mouseDragged (int x, int y ) [virtual]
Call-back function for mouse drag.
Reimplemented from BaseGfxApp.
7.18.4.11 void Simulation::pause (int)
7.18.4.12 void Simulation::reinit (int)
7.18.4.13 void Simulation::resume (int)
7.18.4.14 void Simulation::s_addMovingLightSource (int a) [static]
7.18.4.15 void Simulation::s_addObstacle(int a) [static]
```

```
7.18.4.16 void Simulation::s_addRobot(int a) [static]
7.18.4.17 void Simulation::s_addRobotTarget(int a) [static]
7.18.4.18 void Simulation::s_addStationaryLightSource (int a) [static]
7.18.4.19 void Simulation::s_neuralNetworkFileChanged (int a) [static]
7.18.4.20 void Simulation::s_open (int a) [static]
7.18.4.21 void Simulation::s_pause(int a) [static]
7.18.4.22 void Simulation::s_refreshConfiguration (int a) [static]
7.18.4.23 void Simulation::s_reinit(int a) [static]
7.18.4.24 void Simulation::s_removeAllLightSource(int a) [static]
7.18.4.25 void Simulation::s_removeAllObstacle (int a) [static]
7.18.4.26 void Simulation::s_removeAllRobotTarget(int a) [static]
7.18.4.27 void Simulation::s_removeLightSource (int a) [static]
7.18.4.28 void Simulation::s_removeObstacle(int a) [static]
7.18.4.29 void Simulation::s_removeRobotTarget(int a) [static]
7.18.4.30 void Simulation::s_resume (int a) [static]
7.18.4.31 void Simulation::s_save(int a) [static]
7.18.4.32 void Simulation::s_start(int a) [static]
7.18.4.33 void Simulation::start (int)
7.18.4.34 void Simulation::tryAddMovingLightSource ( )
Adds a moving light to the simulation, or raises an error on failure.
7.18.4.35 void Simulation::tryAddObstacle ( )
Adds a obstacle to the simulation, or raises an error on failure.
7.18.4.36 void Simulation::tryAddRobot ( )
Adds a robot to the simulation, or raises an error on failure.
7.18.4.37 void Simulation::tryAddRobotTarget ( )
Adds a robot and target to the simulation, or raises an error on failure.
```

7.18.4.38 void Simulation::tryAddStationaryLightSource ()

Adds a stationary light to the simulation, or raises an error on failure.

```
7.18.4.39 void Simulation::tryOpen ( )
```

Loads a simulation, or raises an error on failure.

```
7.18.4.40 void Simulation::tryRemoveLightSource ( )
```

Removes a light from the simulation, or raises an error on failure.

```
7.18.4.41 void Simulation::tryRemoveObstacle ( )
```

Removes a obstacle from the simulation, or raises an error on failure.

```
7.18.4.42 void Simulation::tryRemoveRobotTarget ( )
```

Removes a robot and target from the simulation, or raises an error on failure.

```
7.18.4.43 void Simulation::trySave ( )
```

Saves a simulation, or raises an error on failure.

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

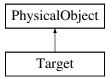
- Simulation.h
- Simulation.cpp

7.19 Target Class Reference

Target for the robot to seek.

```
#include <Target.h>
```

Inheritance diagram for Target:



Public Member Functions

- Target ()
- Target (int radius, Color color, int targetNum=TARGET)
- Target (int maxRadius, int minRadius, Color color, int targetNum=TARGET)
- Target (int radius, Location loc, Color color, int targetNum=TARGET)
- · bool handleCollision (int otherId, bool wasHit)
- ∼Target ()

Additional Inherited Members

7.19.1 Detailed Description

Target for the robot to seek.

7.19.2 Constructor & Destructor Documentation

```
    7.19.2.1 Target::Target ( )
    7.19.2.2 Target::Target ( int radius, Color color, int targetNum = TARGET )
    7.19.2.3 Target::Target ( int maxRadius, int minRadius, Color color, int targetNum = TARGET )
    7.19.2.4 Target::Target ( int radius, Location loc, Color color, int targetNum = TARGET )
```

This is the class destructor.

7.19.2.5 Target:: \sim Target ()

7.19.3 Member Function Documentation

```
7.19.3.1 bool Target::handleCollision (int otherId, bool wasHit) [virtual]
```

Author

Lucas Kramer This function gives Target an appropriate reaction when a collision occurs.

Parameters

otherId	id of the object that it collides with
wasHit	tells if the object was hit previously

Implements PhysicalObject.

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

- Target.h
- Target.cpp

Chapter 8

File Documentation

8.1 artist.cpp File Reference

```
#include "Color.h"
#include "artist.h"
#include "configuration.h"
#include <GL/glut.h>
#include <GL/glu.h>
#include <math.h>
```

Namespaces

· artist

graphical commands

Macros

#define _USE_MATH_DEFINES

Functions

- · void artist::drawObject (Location loc, int radius, Color color)
- void artist::debugArrow (Location loc, int orientation)
- void artist::drawLight (Location loc, int radius, Color color)
- void artist::drawObstacle (Location loc, int radius)
- void artist::drawSensor (Location loc, int orientation, int angle, float intensity)
- void artist::drawRobot (Location loc, int radius, int orientation, Color color, Color lineColor)

8.1.1 Macro Definition Documentation

8.1.1.1 #define _USE_MATH_DEFINES

8.2 artist.h File Reference

A namespace for OpenGL graphical functions.

```
#include "Color.h"
#include "Location.h"
```

Namespaces

· artist

graphical commands

Functions

- void artist::debugArrow (Location loc, int orientation)
- void artist::drawObject (Location loc, int radius, Color color)
- void artist::drawLight (Location loc, int radius, Color color)
- void artist::drawObstacle (Location loc, int radius)
- void artist::drawSensor (Location loc, int orientation, int angle, float intensity)
- void artist::drawRobot (Location loc, int radius, int orientation, Color color, Color lineColor)

8.2.1 Detailed Description

A namespace for OpenGL graphical functions.

Author

Carl Bahn Lucas Kramer

8.3 BaseGfxApp.cpp File Reference

```
Implementation of BaseGfxApp.
#include "BaseGfxApp.h"
```

8.3.1 Detailed Description

Implementation of BaseGfxApp.

Author

CSci3081 Guru Carl Bahn, Lucas Kramer

8.4 BaseGfxApp.h File Reference

The basic application class for CSci-3081 project. Uses GLUT and GLUI and wraps them in a nice C++ interface.

```
#include <string>
#include <iostream>
#include <assert.h>
#include <GL/glui.h>
```

Classes

class BaseGfxApp

Graphics driver class.

8.4.1 Detailed Description

The basic application class for CSci-3081 project. Uses GLUT and GLUI and wraps them in a nice C++ interface.

Author

CSci3081 Guru Carl Bahn, Lucas Kramer

8.5 Color.cpp File Reference

Utility functions for handling color.

```
#include "Color.h"
#include "configuration.h"
#include <iostream>
#include <cstdlib>
#include <math.h>
```

8.5.1 Detailed Description

Utility functions for handling color.

Author

Carl Bahn Lucas Kramer

8.6 Color.h File Reference

Definintions for representing any hexedecimal color.

```
#include <iostream>
```

Classes

• struct Color

This struct holds the representation of a color.

Macros

• #define GET_COLOR(COLOR_VAR)

Typedefs

• typedef struct Color Color

This struct holds the representation of a color.

8.6.1 Detailed Description

Definintions for representing any hexedecimal color.

Author

Lucas Kramer

8.6.2 Macro Definition Documentation

```
8.6.2.1 #define GET_COLOR( COLOR_VAR )
```

Value:

8.6.3 Typedef Documentation

8.6.3.1 typedef struct Color Color

This struct holds the representation of a color.

Each field is a float value that shoult be between 0 and 1. This allows a direct translation from hex colors, with each field being the hex RGB value / 255.

8.7 ComplexRobot.cpp File Reference

The representation of a robot within the simulation.

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdexcept>
#include "environment.h"
#include "Robot.h"
#include "ComplexRobot.h"
#include "configuration.h"
```

8.7.1 Detailed Description

The representation of a robot within the simulation.

Author

Lucas Kramer

8.8 ComplexRobot.h File Reference

The representation of a robot within the simulation.

```
#include "PhysicalObject.h"
#include "Robot.h"
#include "Sensor.h"
#include "configuration.h"
```

Classes

class ComplexRobot

A complex robot with configurable feedback from all sensors.

8.8.1 Detailed Description

The representation of a robot within the simulation.

Author

Lucas Kramer

8.9 configuration.cpp File Reference

Implementation for configuration file loader.

```
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <string.h>
#include <boost/regex.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/regex.hpp>
#include "configuration.h"
```

Typedefs

 typedef unordered_map< string, configValue > ConfigTable

A type alias for an unordered map containing strings mapped to configValues.

Functions

• bool isAllWhitespace (const string &line)

A helper function for loadConfig that tests if the line is valid input.

- configValue parseValue (const string &type, const string &valueText)
- ConfigTable mergeConfigTables (ConfigTable c1, ConfigTable c2)
- ConfigTable loadConfig (const string &filename)
- string expandVar (const smatch &match)

8.9.1 Detailed Description

Implementation for configuration file loader.

Author

Lucas Kramer Carl Bahn

See config.h for more information.

8.9.2 Typedef Documentation

8.9.2.1 typedef unordered_map<string, configValue> ConfigTable

A type alias for an unordered map containing strings mapped to configValues.

8.9.3 Function Documentation

```
8.9.3.1 string expandVar (const smatch & match)
```

8.9.3.2 bool is All Whitespace (const string & line)

A helper function for loadConfig that tests if the line is valid input.

```
8.9.3.3 ConfigTable loadConfig ( const string & filename )
```

- 8.9.3.4 ConfigTable mergeConfigTables (ConfigTable c1, ConfigTable c2)
- 8.9.3.5 configValue parseValue (const string & type, const string & valueText)

8.10 configuration.h File Reference

Interface definitions for configuration file loader.

```
#include <string>
#include <unordered_map>
```

Classes

· struct configValue

This struct holds a configuration value of any legal type.

· class Configuration

Macros

- #define GET_INT(NAME) Configuration::get()->getIntConfig(NAME)
- #define GET_FLOAT(NAME) Configuration::get()->getFloatConfig(NAME)
- #define GET_BOOL(NAME) Configuration::get()->getBoolConfig(NAME)
- #define GET_CHAR(NAME) Configuration::get()->getCharConfig(NAME)
- #define GET_STRING(NAME) Configuration::get()->getStringConfig(NAME)

8.10.1 Detailed Description

Interface definitions for configuration file loader.

Author

Lucas Kramer

The format of the config file is a series of lines with the format

<type> <name> = <value>

Blank lines are allowed, and comments # can occur at the end of a line.

Valid types are int, hex, octal, float, bool/boolean, char, and string. Note that int, hex, and octal all create a variable with type int, and only affect how the value is parsed.

Names can be [A-Za-z][A-Za-z0-9_-]*, but the recommended usage is all upper case, seperated by underscores.

The following is a list of valid value formats:

Type name	Format	Examples	Notes
int	[0-9]+	42	
hex	(0x)?[0-9A-Fa-f]+	0x3AF4, 0x3af4, 3AF4	It is recommended to use the 0x prefix and upper case A-F
hex	[0-7]+	0123, 123	It is recommended to use the 0 prefix
float	[0-9]+(.[0-9]+)?	3, 3.14	
bool/boolean	true false	true, false	It is recommended to use the bool type name
string	"[^"\n]*"	"Hello, World! ", "", "\$OTHER"	\$ followed by a variable is expanded to that variable. An actual \$ must be escaped
char	'([^\n\r\t] \\n \\r \\t)'	'a', 'A', '4', '\n', '\t'	

In addition to variable settings, it is possible to load other configuration files. This has the syntax:

use <filename>

where filename is a string. The path to <filename> is with respect to the directory containing the current file - absolute paths are not permitted. When an included file contains a variable with the same name as a value that is already defined, the current behavior is to overwrite the old value. Any value re-definition causes a warning.

8.10.2 Macro Definition Documentation

 $8.10.2.1 \quad \text{\#define GET_BOOL(} \quad \textit{NAME} \text{) } \textbf{Configuration::get()->getBoolConfig(NAME)}$

8.10.2.2 #define GET_CHAR(NAME) Configuration::get()->getCharConfig(NAME)

 $8.10.2.3 \quad \text{\#define GET_FLOAT(} \quad \textit{NAME} \text{) Configuration::get()->getFloatConfig(NAME)}$

8.10.2.4 #define GET_INT(NAME) Configuration::get()->getIntConfig(NAME)

8.10.2.5 #define GET_STRING(NAME) Configuration::get()->getStringConfig(NAME)

8.11 environment.cpp File Reference

```
#include "PhysicalObject.h"
#include "environment.h"
#include "configuration.h"
#include <iostream>
#include <math.h>
```

Functions

• bool isTouching (Location I1, int r1, Location I2, int r2)

8.11.1 Function Documentation

```
8.11.1.1 bool isTouching (Location 11, int r1, Location 12, int r2)
```

Checks if two objects specified by their Locations and radii

Returns

a boolean value that is true when objects are overlapping

8.12 environment.h File Reference

```
#include "Location.h"
#include "configuration.h"
#include <unordered_map>
#include <vector>
#include <mutex>
#include <stdexcept>
```

Classes

· class ObjectIterator

ObjectIterator class, a simple iterator for the object in environment.

class NoOpenLocationException

An exception to be thrown when an open Location cannot be found.

Namespaces

environment

environment namespace, handles all the objects as a group. Also manages object ids and collision detection

Typedefs

· typedef ObjectIterator environment::objectIterator

Functions

int environment::addObject (PhysicalObject *object)

Adds an object to the environment.

void environment::removeObject (int id)

Removes an object from the environment.

void environment::clear ()

Removes all objects from the environment and resets the id counter.

unsigned environment::getNumObjects ()

Gets the number of objects in environment.

PhysicalObject * environment::getObject (int id)

Finds an object from the environment.

· objectIterator environment::getObjectsBegin ()

Gets an iterator to the beginning of the objects.

• objectIterator environment::getObjectsEnd ()

Gets an iterator to the end of the objects.

- bool environment::isTouchingWall (Location I, int r)
- bool environment::isTouchingWall (int id)
- bool environment::isTouchingObject (Location I, int r, int id)
- bool environment::isTouchingHitableObject (Location I, int r, int id)
- bool environment::isTouchingObject (Location I, int r)
- bool environment::isTouchingHitableObject (Location I, int r)
- bool environment::isTouchingObject (int id)
- bool environment::isTouchingHitableObject (int id)
- bool environment::isColliding (Location I, int r)
- bool environment::isColliding (int id)
- bool environment::isCollidingWithHitable (Location I, int r)
- · bool environment::isCollidingWithHitable (int id)
- int environment::getCollisionId (Location I, int r, int id)
- int environment::getHitableCollisionId (Location I, int r, int id)
- int environment::getCollisionId (Location I, int r)
- int environment::getHitableCollisionId (Location I, int r)
- int environment::getCollisionId (int id)
- · int environment::getHitableCollisionId (int id)

8.13 LightSource.cpp File Reference

The representation of a target in the simulation.

```
#include <stdlib.h>
#include <math.h>
#include "artist.h"
#include "environment.h"
#include "LightSource.h"
#include "configuration.h"
```

8.13.1 Detailed Description

The representation of a target in the simulation.

Author

Himawan Go Lucas Kramer

8.14 LightSource.h File Reference

The representation of an obstacle in the simulation.

```
#include "PhysicalObject.h"
#include "configuration.h"
```

Classes

· class LightSource

LightSource for the robot to seek.

8.14.1 Detailed Description

The representation of an obstacle in the simulation.

Author

Himawan Go Lucas Kramer

8.15 Location.h File Reference

Classes

struct Location

an xy position on the screen, approximately in pixels x is horizontal from the left and y is vertical from the bottom

Typedefs

• typedef struct Location Location

an xy position on the screen, approximately in pixels x is horizontal from the left and y is vertical from the bottom

8.15.1 Typedef Documentation

8.15.1.1 typedef struct Location Location

an xy position on the screen, approximately in pixels x is horizontal from the left and y is vertical from the bottom

8.16 main.cpp File Reference

The main function to execute the simulation program.

```
#include <stdlib.h>
#include <time.h>
#include "OptimizeSimulation.h"
#include "Simulation.h"
#include "configuration.h"
```

Macros

• #define DEFAULT_CONFIG "../config/default"

Functions

• int main (int argc, char *argv[])

8.16.1 Detailed Description

The main function to execute the simulation program.

Author

```
Lucas Kramer
Carl Bahn
Himawan Go
```

8.16.2 Macro Definition Documentation

```
8.16.2.1 #define DEFAULT_CONFIG "../config/default"
```

8.16.3 Function Documentation

```
8.16.3.1 int main ( int argc, char * argv[] )
```

Main function to execute the simulation

8.17 mainpage.h File Reference

8.18 NeuralNetwork.cpp File Reference

A feed-forward neural network class.

```
#include <stdlib.h>
#include <iostream>
#include <vector>
#include <mutex>
#include <fstream>
#include <sstream>
#include <string>
#include <string>
#include <boost/regex.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/regex.hpp>
#include "NeuralNetwork.h"
```

8.18.1 Detailed Description

A feed-forward neural network class.

Author

Lucas Kramer

8.19 NeuralNetwork.h File Reference

A feed-forward neural network class.

```
#include <vector>
#include <mutex>
```

Classes

· class NeuralNetwork

A representation of a neural network.

8.19.1 Detailed Description

A feed-forward neural network class.

Author

Lucas Kramer

8.20 NeuralNetworkRobot.cpp File Reference

The representation of a neural network robot within the simulation.

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdexcept>
#include "environment.h"
#include "Robot.h"
#include "NeuralNetworkRobot.h"
#include "NeuralNetwork.h"
#include "configuration.h"
```

8.20.1 Detailed Description

The representation of a neural network robot within the simulation.

Author

Lucas Kramer

8.21 NeuralNetworkRobot.h File Reference

The representation of a neural network robot.

```
#include "PhysicalObject.h"
#include "Robot.h"
#include "Sensor.h"
#include "NeuralNetwork.h"
#include "configuration.h"
```

Classes

· class NeuralNetworkRobot

A robot controlled by a neural network with inputs from the sensors.

8.21.1 Detailed Description

The representation of a neural network robot.

Author

Lucas Kramer

8.22 Obstacle.cpp File Reference

The representation of an obstacle in the simulation.

```
#include <stdlib.h>
#include <math.h>
#include "environment.h"
#include "Obstacle.h"
#include "configuration.h"
#include "artist.h"
```

8.22.1 Detailed Description

The representation of an obstacle in the simulation.

Author

Himawan Go Lucas Kramer

8.23 Obstacle.h File Reference

The representation of an obstacle in the simulation.

```
#include "PhysicalObject.h"
#include "configuration.h"
```

Classes

class Obstacle

Obstacle for the robot to hit/avoid.

8.23.1 Detailed Description

The representation of an obstacle in the simulation.

Author

Himawan Go Lucas Kramer

8.24 OptimizeSimulation.cpp File Reference

```
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <algorithm>
#include <vector>
#include <mutex>
#include <stdexcept>
#include <sstream>
#include <fstream>
#include <iostream>
#include <climits>
#include <sys/stat.h>
#include <boost/interprocess/sync/named_upgradable_mutex.hpp>
#include "Robot.h"
#include "SimpleRobot.h"
#include "ComplexRobot.h"
#include "NeuralNetworkRobot.h"
#include "NeuralNetwork.h"
#include "Target.h"
#include "Obstacle.h"
#include "LightSource.h"
#include "Location.h"
#include "configuration.h"
#include "environment.h"
#include "util.h"
#include "OptimizeSimulation.h"
```

8.25 OptimizeSimulation.h File Reference

Robot simultaion for optimizing the NeuralNetworkRobot.

```
#include <vector>
#include doost/interprocess/sync/named_upgradable_mutex.hpp>
#include "Color.h"
#include "Location.h"
#include "configuration.h"
#include "environment.h"
#include "util.h"
#include "PhysicalObject.h"
#include "NeuralNetwork.h"
```

Classes

class OptimizeSimulation

OptimizeSimulation class, sets up environments and robots.

8.25.1 Detailed Description

Robot simultaion for optimizing the NeuralNetworkRobot.

Author

Lucas Kramer

8.26 PhysicalObject.cpp File Reference

The representation of any object within the simulation.

```
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <assert.h>
#include <unistd.h>
#include <stdexcept>
#include "artist.h"
#include "PhysicalObject.h"
#include "environment.h"
#include "configuration.h"
```

Macros

• #define _USE_MATH_DEFINES

8.26.1 Detailed Description

The representation of any object within the simulation.

Author

Lucas Kramer Himawan Go, Carl Bahn

8.26.2 Macro Definition Documentation

```
8.26.2.1 #define _USE_MATH_DEFINES
```

8.27 PhysicalObject.h File Reference

The representation of any object within the simulation.

```
#include "Color.h"
#include "configuration.h"
#include "Location.h"
```

Classes

· class PhysicalObject

This is a common superclass for all objects.

Enumerations

```
    enum ObjectType {
        ROBOT, TARGET, OBSTACLE, LIGHT,
        LAST = LIGHT }
```

8.27.1 Detailed Description

The representation of any object within the simulation.

Author

Lucas Kramer Himawan Go, Carl Bahn

8.27.2 Enumeration Type Documentation

```
8.27.2.1 enum ObjectType
```

Enumerator

ROBOT

TARGET

OBSTACLE

LIGHT

LAST

8.28 Robot.cpp File Reference

The representation of robot within the simulation.

```
#include <stdexcept>
#include <math.h>
#include "artist.h"
#include "environment.h"
#include "Robot.h"
#include "configuration.h"
```

8.28.1 Detailed Description

The representation of robot within the simulation.

Author

Lucas Kramer Himawan Go, Carl Bahn

8.29 Robot.h File Reference

The representation of robot within the simulation.

```
#include "PhysicalObject.h"
#include "Sensor.h"
#include "configuration.h"
```

Classes

· class Robot

Robot that moves around the window and bumps into obstacles.

Enumerations

enum RobotType { SIMPLE, COMPLEX, NEURAL_NETWORK }

8.29.1 Detailed Description

The representation of robot within the simulation.

Author

Himawan Go Lucas Kramer

8.29.2 Enumeration Type Documentation

```
8.29.2.1 enum RobotType
```

Enumerator

SIMPLE

COMPLEX

NEURAL_NETWORK

8.30 Sensor.cpp File Reference

A sensor that detects certain objects in the environment.

```
#include "artist.h"
#include "Sensor.h"
#include "environment.h"
#include "configuration.h"
#include "PhysicalObject.h"
#include <iostream>
#include <math.h>
#include <stdexcept>
```

Macros

• #define _USE_MATH_DEFINES

94 File Documentation

8.30.1 Detailed Description

A sensor that detects certain objects in the environment.

Author

Himawan Go Carl Bahn Lucas Kramer

8.30.2 Macro Definition Documentation

```
8.30.2.1 #define _USE_MATH_DEFINES
```

8.31 Sensor.h File Reference

A sensor that detects certain objects in the environment.

```
#include "Color.h"
#include "Location.h"
#include "configuration.h"
#include "PhysicalObject.h"
```

Classes

class Sensor

8.31.1 Detailed Description

A sensor that detects certain objects in the environment.

Author

Himawan Go Carl Bahn

8.32 SimpleRobot.cpp File Reference

The representation of robot within the simulation.

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdexcept>
#include "environment.h"
#include "Robot.h"
#include "SimpleRobot.h"
#include "configuration.h"
```

8.32.1 Detailed Description

The representation of robot within the simulation.

Author

Lucas Kramer

8.33 SimpleRobot.h File Reference

The representation of a simple robot.

```
#include "PhysicalObject.h"
#include "Robot.h"
#include "Sensor.h"
#include "configuration.h"
```

Classes

class SimpleRobot

A simple robot with uncrossed feedback.

8.33.1 Detailed Description

The representation of a simple robot.

Author

Lucas Kramer

8.34 Simulation.cpp File Reference

Implementation for the main application class of the robot simulation.

```
#include <unistd.h>
#include <limits.h>
#include <deque>
#include <mutex>
#include <stdexcept>
#include <sstream>
#include <iostream>
#include <fstream>
#include <dirent.h>
#include <sys/stat.h>
#include "PhysicalObject.h"
#include "Robot.h"
#include "SimpleRobot.h"
#include "ComplexRobot.h"
#include "NeuralNetworkRobot.h"
#include "Target.h"
#include "Obstacle.h"
#include "LightSource.h"
#include "Location.h"
#include "configuration.h"
#include "environment.h"
#include "util.h"
#include "Simulation.h"
```

Functions

· string ftos (float f)

96 File Documentation

8.34.1 Detailed Description

Implementation for the main application class of the robot simulation.

Author

```
Lucas Kramer
Lucas Kramer Xi Zhang, Carl Bahn
```

8.34.2 Function Documentation

```
8.34.2.1 string ftos ( float f )
```

8.35 Simulation.h File Reference

Main application class for the robot simulation.

```
#include "Color.h"
#include "Location.h"
#include "configuration.h"
#include "environment.h"
#include "util.h"
#include "BaseGfxApp.h"
#include "PhysicalObject.h"
```

Classes

class Simulation

Simulation class, sets up the GUI and the drawing environment.

8.35.1 Detailed Description

Main application class for the robot simulation.

Author

Lucas Kramer Xi Zhang, Carl Bahn

8.36 Target.cpp File Reference

The representation of a target in the simulation.

```
#include <stdlib.h>
#include <math.h>
#include "environment.h"
#include "Target.h"
#include "configuration.h"
```

8.36.1 Detailed Description

The representation of a target in the simulation.

Author

Himawan Go Lucas Kramer

8.37 Target.h File Reference

The representation of an obstacle in the simulation.

```
#include "PhysicalObject.h"
```

Classes

· class Target

Target for the robot to seek.

8.37.1 Detailed Description

The representation of an obstacle in the simulation.

Author

Himawan Go Lucas Kramer

8.38 util.cpp File Reference

Implmentation of functions for the simulation.

```
#include <unistd.h>
#include <stack>
#include <queue>
#include <mutex>
#include <stdexcept>
#include <sstream>
#include <iostream>
#include <fstream>
#include <GL/glui.h>
#include <assert.h>
#include "Robot.h"
#include "SimpleRobot.h"
#include "ComplexRobot.h"
#include "NeuralNetwork.h"
#include "NeuralNetworkRobot.h"
#include "Target.h"
#include "Obstacle.h"
#include "LightSource.h"
#include "Location.h"
#include "configuration.h"
#include "environment.h"
#include "util.h"
```

Variables

• std::mutex objectsMutex

98 File Documentation

- stack< int > robots
- stack < int > targets
- stack< int > lights
- stack< int > obstacles
- int colorNum = 0

8.38.1 Detailed Description

Implmentation of functions for the simulation.

Author

Lucas Kramer

8.38.2 Variable Documentation

```
8.38.2.1 int colorNum = 0
```

8.38.2.2 stack<int> lights

8.38.2.3 std::mutex objectsMutex

8.38.2.4 stack<int> obstacles

8.38.2.5 stack<int> robots

8.38.2.6 stack<int> targets

8.39 util.h File Reference

Helper functions for the simulation.

```
#include "NeuralNetwork.h"
```

Namespaces

util

util namespace, contains helper functions to add and remove robots

Functions

· void util::reset ()

Removes all objects and resets to the initial state.

• void util::display ()

Function to render all the objects. This function is called repeatedly from the simulation, and iterates through the objects and calls their respective display functions.

· void util::advance ()

Function to update the positions of all objects. This function is called repeatedly from the simulation, and iterates through the objects and calls their respective updatePosition functions.

Color util::newColor ()

Gets a new, unused color.

• int util::getNumRobotsTargets ()

8.39 util.h File Reference 99

Gets the number of robots/target pairs.

int util::getNumLights ()

Gets the number of lights.

• int util::getNumObstacles ()

Gets the number of obstacles.

bool util::addRobotTarget (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robotSensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetworkFile)

Adds a robot and paired target to the simulation.

bool util::addRobot (int robotType, int lightSensorConnectionPattern, int robotSensorConnectionPattern, int obstacleSensorConnectionPattern, int targetSensorConnectionPattern, float lightSensorScale, float robot-SensorScale, float obstacleSensorScale, float targetSensorScale, int initialSpeed, std::string neuralNetwork-File)

Adds a robot to the simulation.

bool util::addNeuralNetworkRobotTarget (const NeuralNetwork &network)

Adds a neural network robot and a paired target to the simulation.

• bool util::addStationaryLightSource ()

Adds a light to the simulation.

· bool util::addMovingLightSource ()

Adds a light to the simulation.

• bool util::addObstacle ()

Adds a obstacle to the simulation.

bool util::copy (int id, Location loc)

Copies an object to a new Location.

• bool util::removeRobotTarget ()

Removes a robot from the simulation.

• bool util::removeLightSource ()

Removes a light from the simulation.

• bool util::removeObstacle ()

Removes a obstacle from the simulation.

void util::removeAllRobotTarget ()

Removes all robots.

void util::removeAllLightSource ()

Removes all lights.

• void util::removeAllObstacle ()

Removes all obstacles.

• bool util::open (std::string filename)

Loads a simulation from a file.

bool util::save (std::string filename)

Saves the current state to a file.

8.39.1 Detailed Description

Helper functions for the simulation.

Author

Lucas Kramer

Index

Dana Ofri Ann	duay. Ohio et 10	
~BaseGfxApp	drawObject, 13	
BaseGfxApp, 30	drawObstacle, 14	
~ComplexRobot	drawRobot, 14	
ComplexRobot, 36	drawSensor, 14	
~LightSource	artist.cpp, 77	
LightSource, 42	_USE_MATH_DEFINES, 77	
~NeuralNetworkRobot	artist.h, 77	
NeuralNetworkRobot, 47	Dana Chu Anna 00	
~ObjectIterator	BaseGfxApp, 29	
ObjectIterator, 49	~BaseGfxApp, 30	
~Obstacle	advance, 30	
Obstacle, 51	BaseGfxApp, 30	
\sim OptimizeSimulation	BaseGfxApp, 30	
OptimizeSimulation, 52	display, 30	
\sim PhysicalObject	glui, <mark>30</mark>	
PhysicalObject, 54	gluiControl, 31	
\sim Robot	graphicsBegin, 31	
Robot, 61	graphicsTimer, 31	
\sim SimpleRobot	handle, 31	
SimpleRobot, 68	height, 31	
\sim Simulation	keyboard, 31	
Simulation, 71	keyboardSpecial, 31	
\sim Target	keyboardSpecialUp, 31	
Target, 75	keyboardUp, 31	
_USE_MATH_DEFINES	leftMouseDown, 31	
artist.cpp, 77	leftMouseUp, 31	
PhysicalObject.cpp, 91	m_drag, 32	
Sensor.cpp, 94	m_glui, 32	
	m_glutWindowHandle, 32	
addMovingLightSource	m_height, 32	
util, 23	m_width, 32	
addNeuralNetworkRobotTarget	middleMouseDown, 31	
util, 23	middleMouseUp, 31	
addObject	mouseDragged, 31	
environment, 15	mouseMoved, 31	
addObstacle	reshape, 32	
util, 24	rightMouseDown, 32	
addRobot	rightMouseUp, 32	
util, 24	runMainLoop, 32	
addRobotTarget	s currentApp, 32	
util, 24	s draw, 32	
addStationaryLightSource	s_gluicallback, 32	
util, 25	s_glutInitialized, 32	
advance	s keyboard, 32	
BaseGfxApp, 30	s_keyboardspecial, 32	
Simulation, 71	s keyboardspecialup, 32	
util, 25	s_keyboardup, 32	
artist, 13	s_mousebtn, 32	
debugArrow, 13	s_mousemotion, 32	
drawLight, 13	s reshape. 32	

setCaption, 32	charVal, 40
width, 32	floatVal, 40
BaseGfxApp.cpp, 78	intVal, 40
BaseGfxApp.h, 78	stringVal, 40
blue	type, 40
Color, 35	Configuration, 37
boolVal	get, <mark>38</mark>
configValue, 40	getBoolConfig, 38
OOMBLEY.	getCharConfig, 38
COMPLEX	getFloatConfig, 38
Robot.h, 93	getIntConfig, 39
charVal	getStringConfig, 39
configValue, 40	initConfig, 39
clear	refresh, 39
environment, 16	configuration.cpp, 81
Color, 33	ConfigTable, 82
blue, 35	expandVar, 82
Color, 33, 34	isAllWhitespace, 82
Color.h, 80	loadConfig, 82
green, 35	mergeConfigTables, 82
isSimilar, 34	parseValue, 82
operator<<, 35	configuration.h, 82
operator==, 35	GET_BOOL, 83
red, 35	GET_CHAR, 83
Color.cpp, 79	GET_FLOAT, 83
Color.h, 79	GET_INT, 83
Color, 80	GET_STRING, 83
GET_COLOR, 80	сору
colorNum	util, 25
util.cpp, 98	
combine	DEFAULT_CONFIG
NeuralNetwork, 44	main.cpp, 87
ComplexRobot, 35	debugArrow
∼ComplexRobot, 36	artist, 13
ComplexRobot, 36	defaultSpeed
ComplexRobot, 36	ComplexRobot, 37
defaultSpeed, 37	display
enableLightSensors, 37	BaseGfxApp, 30
enableObstacleSensors, 37	LightSource, 42
enableRobotSensors, 37	PhysicalObject, 54
enableTargetSensors, 37	Robot, 62
getLeftSpeed, 37	Sensor, 64
getRightSpeed, 37	Simulation, 71
lightSensorScale, 37	util, 25
lightSensorsCrossed, 37	drawLight
obstacleSensorScale, 37	artist, 13
obstacleSensorsCrossed, 37	drawObject
robotSensorScale, 37	artist, 13
robotSensorsCrossed, 37	drawObstacle
targetSensorScale, 37	artist, 14
targetSensorsCrossed, 37	drawRobot
ComplexRobot.cpp, 80	artist, 14
ComplexRobot.h, 80	drawSensor
compute	artist, 14
NeuralNetwork, 44	
ConfigTable	enableLightSensors
configuration.cpp, 82	ComplexRobot, 37
configValue, 39	enableObstacleSensors
boolVal, 40	ComplexRobot, 37

enableRobotSensors	getCharConfig		
ComplexRobot, 37	Configuration, 38		
enableTargetSensors	getCollisionId		
ComplexRobot, 37	environment, 16		
environment, 14	getColor		
addObject, 15	PhysicalObject, 56		
clear, 16	getFloatConfig		
getCollisionId, 16	Configuration, 38		
getHitableCollisionId, 17	getHitableCollisionId		
getNumObjects, 18	environment, 17		
getObject, 18	getld		
getObjectsBegin, 18	PhysicalObject, 56		
getObjectsEnd, 18	getIntConfig		
isColliding, 18, 19	Configuration, 39		
isCollidingWithHitable, 19	getLeftSpeed		
isTouchingHitableObject, 20	ComplexRobot, 37		
isTouchingObject, 20, 21	NeuralNetworkRobot, 47		
isTouchingWall, 21	Robot, 62		
objectIterator, 15	SimpleRobot, 68		
removeObject, 22	getLineColor		
environment.cpp, 84	Robot, 62		
isTouching, 84	getLocation		
environment.h, 84	PhysicalObject, 56		
environment::getObjectsBegin	getNumLights		
ObjectIterator, 49	util, 26		
environment::getObjectsEnd	getNumObjects		
ObjectIterator, 49	environment, 18		
expandVar	getNumObstacles		
configuration.cpp, 82	util, 26		
	getNumRobotsTargets		
filename	util, 26		
NeuralNetworkRobot, 47	getObject		
findOpenLocation	environment, 18		
PhysicalObject, 54	getObjectsBegin		
floatVal	environment, 18		
configValue, 40	getObjectsEnd		
forceSetPosition	environment, 18		
PhysicalObject, 54	getOrientation		
forceTranslate	PhysicalObject, 56		
PhysicalObject, 54	Sensor, 64		
ftos	getPerformance		
Simulation.cpp, 96	OptimizeSimulation, 52		
OFT DOOL	getPerformanceMaze		
GET_BOOL	_		
configuration.h, 83	OptimizeSimulation, 52		
GET_CHAR	getPerformanceObstacles		
configuration.h, 83	OptimizeSimulation, 52		
GET_COLOR	getPerformanceRandomRepeated		
Color.h, 80	Orationing Circuitation, FO		
GET_FLOAT	OptimizeSimulation, 52		
	getPosition		
configuration.h, 83	getPosition Sensor, 64		
GET_INT	getPosition Sensor, 64 getRadius		
GET_INT configuration.h, 83	getPosition Sensor, 64 getRadius PhysicalObject, 56		
GET_INT configuration.h, 83 GET_STRING	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed		
GET_INT configuration.h, 83 GET_STRING configuration.h, 83	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed ComplexRobot, 37		
GET_INT configuration.h, 83 GET_STRING configuration.h, 83 get	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed ComplexRobot, 37 NeuralNetworkRobot, 47		
GET_INT configuration.h, 83 GET_STRING configuration.h, 83 get Configuration, 38	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed ComplexRobot, 37 NeuralNetworkRobot, 47 Robot, 62		
GET_INT configuration.h, 83 GET_STRING configuration.h, 83 get Configuration, 38 getBoolConfig	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed ComplexRobot, 37 NeuralNetworkRobot, 47 Robot, 62 SimpleRobot, 68		
GET_INT configuration.h, 83 GET_STRING configuration.h, 83 get Configuration, 38	getPosition Sensor, 64 getRadius PhysicalObject, 56 getRightSpeed ComplexRobot, 37 NeuralNetworkRobot, 47 Robot, 62		

PhysicalObject, 56	environment, 20
getStringConfig	isTouchingObject
Configuration, 39	environment, 20, 21
getTarget	isTouchingWall
Robot, 62	environment, 21
getTypeDetected	
Sensor, 65	keyboard
getViewAngle	BaseGfxApp, 31
Sensor, 65	Simulation, 72
getXPosition	keyboardSpecial
PhysicalObject, 56	BaseGfxApp, 31
Sensor, 65	Simulation, 72
getYPosition	keyboardSpecialUp
PhysicalObject, 57	BaseGfxApp, 31
Sensor, 65	keyboardUp
glui	BaseGfxApp, 31
BaseGfxApp, 30	
gluiControl	LAST
BaseGfxApp, 31	PhysicalObject.h, 92
Simulation, 71	LIGHT
graphicsBegin	PhysicalObject.h, 92
BaseGfxApp, 31	leftMouseDown
graphicsTimer	BaseGfxApp, 31
BaseGfxApp, 31	Simulation, 72
green	leftMouseUp
Color, 35	BaseGfxApp, 31
00101, 00	Simulation, 72
handle	lightSensorScale
BaseGfxApp, 31	ComplexRobot, 37
handleCollision	lightSensorsCrossed
LightSource, 42	ComplexRobot, 37
Obstacle, 51	LightSource, 40
PhysicalObject, 57	~LightSource, 42
Robot, 62	display, 42
Target, 75	handleCollision, 42
hasEqualPosition	LightSource, 41, 42
PhysicalObject, 57	LightSource, 41, 42
	LightSource.cpp, 85
height Page Cfv App. 31	LightSource.h, 86
BaseGfxApp, 31	lights
initConfig	
Configuration, 39	util.cpp, 98 load
initObjects	
Simulation, 72	NeuralNetwork, 44
intVal	loadConfig
configValue, 40	configuration.cpp, 82
isAllWhitespace	Location, 43
configuration.cpp, 82	Location, 43
	Location.h, 86
isColliding	operator==, 43
environment, 18, 19	x, 43
isCollidingWithHitable	y, 43
environment, 19	Location.h, 86
isHitable	Location, 86
PhysicalObject, 60	
isSimilar	m_drag
Color, 34	BaseGfxApp, 32
isTouching	m_glui
environment.cpp, 84	BaseGfxApp, 32
isTouchingHitableObject	m_glutWindowHandle

Description 00	Object to see a 40
BaseGfxApp, 32	ObjectIterator, 49
m_height	ObjectIterator, 49
BaseGfxApp, 32	operator<, 49
m_width	operator>, 49
BaseGfxApp, 32	operator*, 49
main	operator++, 49
main.cpp, 87	operator==, 49
main.cpp, 86	objectIterator
DEFAULT_CONFIG, 87	environment, 15
main, 87	ObjectType
mainpage.h, 87	PhysicalObject.h, 92
mergeConfigTables	objectType
configuration.cpp, 82	PhysicalObject, 60
middleMouseDown	objectsMutex
BaseGfxApp, 31	util.cpp, 98
Simulation, 72	Obstacle, 50
middleMouseUp	∼Obstacle, 51
BaseGfxApp, 31	handleCollision, 51
mouseDragged	Obstacle, 50, 51
BaseGfxApp, 31	Obstacle.cpp, 89
Simulation, 72	Obstacle.h, 89
mouseMoved	obstacleSensorScale
BaseGfxApp, 31	ComplexRobot, 37
mutate	obstacleSensorsCrossed
NeuralNetwork, 46	ComplexRobot, 37
NEUDAL NETWODY	obstacles
NEURAL_NETWORK	util.cpp, 98
Robot.h, 93	open
NeuralNetwork, 43	util, 26
combine, 44	operator<
compute, 44	ObjectIterator, 49
load, 44	operator<<
mutate, 46	Color, 35
NeuralNetwork, 44	operator>
NeuralNetwork, 44	ObjectIterator, 49
write, 46	operator*
NeuralNetwork.cpp, 87	ObjectIterator, 49
NeuralNetwork.h, 88	operator++
NeuralNetworkRobot, 46	ObjectIterator, 49
~NeuralNetworkRobot, 47	operator==
filename, 47	Color, 35
getLeftSpeed, 47	Location, 43
getRightSpeed, 47	ObjectIterator, 49
NeuralNetworkRobot, 47	
NeuralNetworkRobot, 47	OptimizeSimulation, 52
NeuralNetworkRobot.cpp, 88	~OptimizeSimulation, 52
•••	getPerformance, 52
NeuralNetworkRobot.h, 88	getPerformanceMaze, 52
newColor	getPerformanceObstacles, 52
util, 26	getPerformanceRandomRepeated, 52
NoOpenLocationException, 48	OptimizeSimulation, 52
NoOpenLocationException, 48	OptimizeSimulation, 52
NoOpenLocationException, 48	runMainLoop, 52
	OptimizeSimulation.cpp, 90
OBSTACLE	OptimizeSimulation.h, 90
PhysicalObject.h, 92	·
ObjectIterator, 48	parseValue
\sim ObjectIterator, 49	configuration.cpp, 82
environment::getObjectsBegin, 49	pause
environment::getObjectsEnd, 49	Simulation, 72

PhysicalObject.h	removeLightSource	
LAST, 92	util, 26	
LIGHT, 92	removeObject	
OBSTACLE, 92	environment, 22	
ROBOT, 92	removeObstacle	
TARGET, 92	util, 27	
PhysicalObject, 53	removeRobotTarget	
~PhysicalObject, 54	util, 27	
display, 54	reorient	
findOpenLocation, 54	PhysicalObject, 58	
forceSetPosition, 54	reset	
forceTranslate, 54	util, 27	
getColor, 56	reshape	
getId, 56	BaseGfxApp, 32	
getLocation, 56	resume	
getOrientation, 56	Simulation, 72	
getRadius, 56	rightMouseDown	
getSpeed, 56	BaseGfxApp, 32	
getXPosition, 56	rightMouseUp	
getYPosition, 57	BaseGfxApp, 32	
handleCollision, 57	• •	
	Robot, 60	
hasEqualPosition, 57	~Robot, 61	
isHitable, 60	display, 62	
objectType, 60	getLeftSpeed, 62	
PhysicalObject, 54	getLineColor, 62	
PhysicalObject, 54	getRightSpeed, 62	
pointTo, 57	getTarget, 62	
reorient, 58	handleCollision, 62	
rotate, 58	Robot, 61	
setColor, 58	robotType, 63	
setLocation, 58	setLineColor, 62	
setOrientation, 58	setTarget, 63	
setPosition, 59	update, 63	
setRadius, 59	updateMembers, 63	
setSpeed, 59	Robot.h	
translate, 59	COMPLEX, 93	
update, 59	NEURAL_NETWORK, 93	
updateMembers, 59	SIMPLE, 93	
updatePosition, 59	Robot.cpp, 92	
PhysicalObject.cpp, 91	Robot.h, 93	
PhysicalObject.h, 91	RobotType, 93	
ObjectType, 92	robotSensorScale	
pointTo	ComplexRobot, 37	
PhysicalObject, 57	robotSensorsCrossed	
	ComplexRobot, 37	
ROBOT	RobotType	
PhysicalObject.h, 92	Robot.h, 93	
red	robotType	
Color, 35	Robot, 63	
refresh	robots	
Configuration, 39	util.cpp, 98	
reinit	rotate	
Simulation, 72	PhysicalObject, 58	
removeAllLightSource	runMainLoop	
util, 26	BaseGfxApp, 32	
removeAllObstacle	OptimizeSimulation, 52	
util, 26	Optimizeoimulation, 32	
removeAllRobotTarget	SIMPLE	
util, 26	Robot.h, 93	
	•	

s_addMovingLightSource	s_start		
Simulation, 72	Simulation, 73		
s_addObstacle	save		
Simulation, 72	util, 27		
s_addRobot	sense		
Simulation, 72	Sensor, 65		
s_addRobotTarget	Sensor, 63		
Simulation, 73	display, 64		
s_addStationaryLightSource	getOrientation, 64		
Simulation, 73	getPosition, 64		
s_currentApp	getTypeDetected, 65		
BaseGfxApp, 32	getVlewAngle, 65		
s_draw	getXPosition, 65 getYPosition, 65		
BaseGfxApp, 32 s_gluicallback	sense, 65		
BaseGfxApp, 32	Sensor, 64		
s_glutInitialized	setOrientation, 65		
BaseGfxApp, 32	setPosition, 67		
s_keyboard	setTypeDetected, 67		
BaseGfxApp, 32	setTypeDetected, 67		
s_keyboardspecial	updatePosition, 67		
BaseGfxApp, 32	Sensor.cpp, 93		
s_keyboardspecialup	_USE_MATH_DEFINES, 94		
BaseGfxApp, 32	Sensor.h, 94		
s_keyboardup	setCaption		
BaseGfxApp, 32	BaseGfxApp, 32		
s mousebtn	setColor		
BaseGfxApp, 32	PhysicalObject, 58		
s mousemotion	setLineColor		
BaseGfxApp, 32	Robot, 62		
s_neuralNetworkFileChanged	setLocation		
Simulation, 73	PhysicalObject, 58		
s_open	setOrientation		
Simulation, 73	PhysicalObject, 58		
s pause	Sensor, 65		
Simulation, 73	setPosition		
s_refreshConfiguration	PhysicalObject, 59		
Simulation, 73	Sensor, 67		
s reinit	setRadius		
Simulation, 73	PhysicalObject, 59		
s_removeAllLightSource	setSpeed		
Simulation, 73	PhysicalObject, 59		
s removeAllObstacle	setTarget		
Simulation, 73	Robot, 63		
s_removeAllRobotTarget	setTypeDetected		
Simulation, 73	Sensor, 67		
s removeLightSource	setViewAngle		
Simulation, 73	Sensor, 67		
s removeObstacle	SimpleRobot, 68		
Simulation, 73	~SimpleRobot, 68		
s_removeRobotTarget	getLeftSpeed, 68		
Simulation, 73	getRightSpeed, 68		
s_reshape	SimpleRobot, 68		
BaseGfxApp, 32	SimpleRobot, 68		
s resume	SimpleRobot.cpp, 94		
Simulation, 73	SimpleRobot.h, 95		
s save	Simulation, 69		
Simulation, 73	\sim Simulation, 71		
	,		

advance, 71	ftos, 96
display, 71	Simulation.h, 96
gluiControl, 71	start
initObjects, 72	Simulation, 73
keyboard, 72	stringVal
keyboardSpecial, 72	configValue, 40
leftMouseDown, 72	TAROFT
leftMouseUp, 72	TARGET
middleMouseDown, 72	PhysicalObject.h, 92
mouseDragged, 72	Target, 74
pause, 72	∼Target, 75
reinit, 72	handleCollision, 75
resume, 72	Target, 75
s_addMovingLightSource, 72	Target.cpp, 96
s_addObstacle, 72	Target.h, 97
s_addRobot, 72	targetSensorScale
s_addRobotTarget, 73	ComplexRobot, 37
s_addStationaryLightSource, 73	targetSensorsCrossed
s_neuralNetworkFileChanged, 73	ComplexRobot, 37
s_open, 73	targets
s_pause, 73	util.cpp, 98
s_refreshConfiguration, 73	translate
s_reinit, 73	PhysicalObject, 59
s_removeAllLightSource, 73	tryAddMovingLightSource
s_removeAllObstacle, 73	Simulation, 73
s_removeAllRobotTarget, 73	tryAddObstacle
s_removeLightSource, 73	Simulation, 73
s_removeObstacle, 73	tryAddRobot
s_removeRobotTarget, 73	Simulation, 73
s_resume, 73	tryAddRobotTarget Simulation, 73
s_save, 73	tryAddStationaryLightSource
s_start, 73	Simulation, 73
Simulation, 71	tryOpen
start, 73	Simulation, 74
tryAddMovingLightSource, 73	tryRemoveLightSource
tryAddObstacle, 73	Simulation, 74
tryAddRobot, 73	tryRemoveObstacle
tryAddRobotTarget, 73	Simulation, 74
tryAddStationaryLightSource, 73	tryRemoveRobotTarget
tryOpen, 74	Simulation, 74
tryRemoveLightSource, 74	trySave
tryRemoveObstacle, 74	Simulation, 74
tryRemoveRobotTarget, 74	type
trySave, 74	configValue, 40
UI_ADD_O, 71	
UI_ADD_RT, 71	UI_ADD_O
UI_PAUSE, 71	Simulation, 71
UI_QUIT, 71	UI_ADD_RT
UI_REFRESH_SETTINGS_O, 71 UI_REMOVE_ALL_O, 71	Simulation, 71
UI_REMOVE_ALL_RT, 71	UI_PAUSE
UI_REMOVE_O, 71	Simulation, 71
UI_REMOVE_RT, 71	UI_QUIT Simulation, 71
UI RESET, 71	UI_REFRESH_SETTINGS_O
UI_RESUME, 71	Simulation, 71
UI_START, 71	UI_REMOVE_ALL_O
UIControlType, 71	Simulation, 71
Simulation.cpp, 95	UI_REMOVE_ALL_RT
11/ -	

Simulation, 71	Х	
UI_REMOVE_O		Location, 43
Simulation, 71		
UI_REMOVE_RT	У	
Simulation, 71		Location, 43
UI_RESET		
Simulation, 71		
UI_RESUME		
Simulation, 71		
UI_START		
Simulation, 71		
UIControlType		
Simulation, 71		
update		
PhysicalObject, 59		
Robot, 63		
updateMembers		
PhysicalObject, 59		
Robot, 63		
updatePosition		
PhysicalObject, 59		
Sensor, 67		
util, 22		
addMovingLightSource, 23		
addNeuralNetworkRobotTarget, 23		
addObstacle, 24		
addRobot, 24		
addRobotTarget, 24		
addStationaryLightSource, 25		
advance, 25		
copy, 25		
display, 25		
getNumLights, 26		
getNumObstacles, 26		
getNumRobotsTargets, 26		
newColor, 26		
open, 26		
removeAllLightSource, 26 removeAllObstacle, 26		
removeAllRobotTarget, 26		
removeLightSource, 26		
removeObstacle, 27		
removeRobotTarget, 27		
reset, 27		
save, 27		
util.cpp, 97		
colorNum, 98		
lights, 98		
objectsMutex, 98		
obstacles, 98		
robots, 98		
targets, 98		
util.h, 98		
width		
BaseGfxApp, 32		
write		
NeuralNetwork, 46		