

ICP Project 2023/2024

1.0

Generated by Doxygen 1.10.0

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Namespace Documentation	7
4.1 Ui Namespace Reference	7
5 Class Documentation	9
5.1 AutoRobot Class Reference	9
5.1.1 Detailed Description	14
5.1.2 Member Enumeration Documentation	14
5.1.2.1 anonymous enum	14
5.1.3 Constructor & Destructor Documentation	15
5.1.3.1 AutoRobot()	15
5.1.3.2 ~AutoRobot()	15
5.1.4 Member Function Documentation	15
5.1.4.1 boundingRect()	15
5.1.4.2 doRotationStep()	15
5.1.4.3 fromJSON()	16
5.1.4.4 getCollisionLookAhead()	16
5.1.4.5 getRotationDirection()	16
5.1.4.6 getTargetAngle()	17
5.1.4.7 move()	17
5.1.4.8 paint()	17
5.1.4.9 setCollisionLookAhead()	17
5.1.4.10 setRotationDirection()	18
5.1.4.11 setTargetAngle()	18
5.1.4.12 toJSON()	18
5.1.4.13 type()	18
5.1.4.14 willCollide()	19
5.1.5 Member Data Documentation	19
5.1.5.1 collisionLookAhead	19
5.1.5.2 rotationDirection	19
5.1.5.3 targetAngle	19
5.2 CheckableButton Class Reference	20
5.2.1 Detailed Description	22
5.2.2 Member Enumeration Documentation	22
5.2.2.1 ObjectType	22
5.2.3 Constructor & Destructor Documentation	23

5.2.3.1 CheckableButton()	23
5.2.4 Member Function Documentation	23
5.2.4.1 getOverlay()	23
5.2.4.2 getWidgetPos()	23
5.2.4.3 mouseMoveEvent	24
5.2.4.4 mousePressEvent	24
5.2.4.5 mouseReleaseEvent	24
5.2.4.6 setOverlay()	25
5.2.5 Member Data Documentation	25
5.2.5.1 objType	25
5.2.5.2 overlay	25
5.3 ExpandableButtonWidget Class Reference	25
5.3.1 Detailed Description	28
5.3.2 Constructor & Destructor Documentation	28
5.3.2.1 ExpandableButtonWidget()	28
5.3.3 Member Function Documentation	28
5.3.3.1 collapse()	28
5.3.3.2 expand	29
5.3.3.3 setOverlay()	29
5.3.4 Member Data Documentation	29
5.3.4.1 autoButton	29
5.3.4.2 controlButton	29
5.3.4.3 mainButton	29
5.3.4.4 obstacleButton	30
5.4 ExpButton Class Reference	30
5.4.1 Detailed Description	31
5.4.2 Constructor & Destructor Documentation	31
5.4.2.1 ExpButton()	31
5.4.3 Member Function Documentation	31
5.4.3.1 mousePressEvent	31
5.4.3.2 pressed	33
5.5 GameObject Class Reference	33
5.5.1 Detailed Description	35
5.5.2 Constructor & Destructor Documentation	36
5.5.2.1 GameObject()	36
5.5.2.2 ~GameObject()	36
5.5.3 Member Function Documentation	36
5.5.3.1 getCenter()	36
5.5.3.2 getPos()	36
5.5.3.3 paint()	36
5.5.3.4 rotation()	37
5.5.3.5 setPos()	37

5.5.3.6 setRotation()	37
5.5.3.7 toJSON()	38
5.6 MainWindow Class Reference	38
5.6.1 Detailed Description	42
5.6.2 Constructor & Destructor Documentation	42
5.6.2.1 MainWindow()	42
5.6.2.2 ~MainWindow()	42
5.6.3 Member Function Documentation	42
5.6.3.1 eventFilter	42
5.6.3.2 goLeft	43
5.6.3.3 goRight	43
5.6.3.4 goStraight	43
5.6.3.5 handleItemDoubleClick	43
5.6.3.6 initScene()	44
5.6.3.7 mouseDoubleClickEvent	44
5.6.3.8 mouseMoveEvent	44
5.6.3.9 mousePressEvent	44
5.6.3.10 mouseReleaseEvent	45
5.6.3.11 on_horizontalSlider_valueChanged	45
5.6.3.12 on_pushButton_clicked	45
5.6.3.13 resizeEvent	46
5.6.3.14 saveSimulation	46
5.6.3.15 setupAnimation()	46
5.6.3.16 showEvent	46
5.6.3.17 stopMoving	46
5.6.3.18 stopRotating	47
5.6.3.19 toggleList	47
5.6.3.20 updateAnimation()	47
5.6.4 Member Data Documentation	47
5.6.4.1 expandableWidget	47
5.6.4.2 listWidget	47
5.6.4.3 overlay	48
5.6.4.4 paramWidget	48
5.6.4.5 simulationEngine	48
5.6.4.6 ui	48
5.7 Obstacle Class Reference	48
5.7.1 Detailed Description	51
5.7.2 Constructor & Destructor Documentation	51
5.7.2.1 Obstacle() [1/2]	51
5.7.2.2 Obstacle() [2/2]	52
5.7.2.3 ~Obstacle()	52
5.7.3 Member Function Documentation	52

5.7.3.1 fromJSON()	52
5.7.3.2 getCenter()	52
5.7.3.3 getPos()	53
5.7.3.4 obstacleSepuku	53
5.7.3.5 paint()	53
5.7.3.6 paramsUpdated	53
5.7.3.7 rotation()	54
5.7.3.8 setPos()	54
5.7.3.9 setRotation()	54
5.7.3.10 toJSON()	55
5.8 OverlayWidget Class Reference	55
5.8.1 Detailed Description	58
5.8.2 Constructor & Destructor Documentation	58
5.8.2.1 OverlayWidget()	58
5.8.3 Member Function Documentation	59
5.8.3.1 anchor()	59
5.8.3.2 convertFromRotatedSystem()	59
5.8.3.3 convertToRotatedSystem()	59
5.8.3.4 getTimeConstant()	60
5.8.3.5 navigateTheSea()	60
5.8.3.6 paintEvent()	60
5.8.3.7 setActiveObject()	60
5.8.3.8 setLastMousePos()	61
5.8.3.9 trySetSail()	61
5.8.4 Member Data Documentation	61
5.8.4.1 activeObject	61
5.8.4.2 graphView	61
5.8.4.3 lastMousePos	62
5.8.4.4 offset	62
5.8.4.5 option	62
5.8.4.6 simEng	62
5.9 ParamEditLine Class Reference	62
5.9.1 Detailed Description	63
5.9.2 Constructor & Destructor Documentation	63
5.9.2.1 ParamEditLine()	63
5.9.3 Member Function Documentation	64
5.9.3.1 focusIn	64
5.9.3.2 focusInEvent()	64
5.9.3.3 focusOut	64
5.9.3.4 focusOutEvent()	64
5.10 ParamWidget Class Reference	65
5.10.1 Detailed Description	69

5.10.2 Constructor & Destructor Documentation	69
5.10.2.1 ParamWidget()	69
5.10.3 Member Function Documentation	70
5.10.3.1 disconnectStalkedObject()	70
5.10.3.2 focusIn	70
5.10.3.3 focusOut	70
5.10.3.4 hide()	70
5.10.3.5 setAngle	70
5.10.3.6 setAngleToRotate	71
5.10.3.7 setDetectionDistance	71
5.10.3.8 setDirection	71
5.10.3.9 setRadius	71
5.10.3.10 setSize	71
5.10.3.11 setSpeed	72
5.10.3.12 setUpEditLine()	72
5.10.3.13 show() [1/3]	72
5.10.3.14 show() [2/3]	72
5.10.3.15 show() [3/3]	73
5.10.3.16 stalk() [1/3]	73
5.10.3.17 stalk() [2/3]	73
5.10.3.18 stalk() [3/3]	74
5.10.3.19 stopStalking()	74
5.10.3.20 updateAutoRobot	74
5.10.3.21 updateObstacle	75
5.10.3.22 updateRobot	75
5.10.4 Member Data Documentation	75
5.10.4.1 angle	75
5.10.4.2 angleToRotate	75
5.10.4.3 detectionDistance	75
5.10.4.4 direction	75
5.10.4.5 keepUpdating	75
5.10.4.6 labelAngle	76
5.10.4.7 labelAngleToRotate	76
5.10.4.8 labelDetectionDistance	76
5.10.4.9 labelDirection	76
5.10.4.10 labelRadius	76
5.10.4.11 labelSize	76
5.10.4.12 labelSpeed	76
5.10.4.13 layout	76
5.10.4.14 numberValidator	76
5.10.4.15 radius	77
5.10.4.16 size	77

5.10.4.17 speed	77
5.10.4.18 stalkedObject	77
5.11 PopupSaveWindow Class Reference	77
5.11.1 Detailed Description	78
5.11.2 Constructor & Destructor Documentation	79
5.11.2.1 PopupSaveWindow()	79
5.11.2.2 ~PopupSaveWindow()	79
5.11.3 Member Function Documentation	79
5.11.3.1 getEnteredText()	79
5.11.3.2 onOkClicked	79
5.11.4 Member Data Documentation	79
5.11.4.1 enteredText	79
5.11.4.2 lineEdit	80
5.12 Robot Class Reference	80
5.12.1 Detailed Description	85
5.12.2 Member Enumeration Documentation	85
5.12.2.1 anonymous enum	85
5.12.2.2 RotationDirection	85
5.12.3 Constructor & Destructor Documentation	85
5.12.3.1 Robot()	85
5.12.3.2 ~Robot()	86
5.12.4 Member Function Documentation	86
5.12.4.1 boundingRect()	86
5.12.4.2 fromJSON()	86
5.12.4.3 getAngle()	86
5.12.4.4 getCenter()	86
5.12.4.5 getDirectionVector()	87
5.12.4.6 getMoveSpeed()	87
5.12.4.7 getPos()	87
5.12.4.8 getRadius()	87
5.12.4.9 getRotationSpeed()	87
5.12.4.10 isActive()	88
5.12.4.11 keyPressEvent()	88
5.12.4.12 keyReleaseEvent()	88
5.12.4.13 move()	88
5.12.4.14 paint()	89
5.12.4.15 paramsUpdated	89
5.12.4.16 pos()	89
5.12.4.17 robotSepuku	89
5.12.4.18 rotation()	89
5.12.4.19 setMoveSpeed()	89
5.12.4.20 setPos() [1/2]	90

5.12.4.21 setPos() [2/2]	90
5.12.4.22 setRadius()	90
5.12.4.23 setRotation()	90
5.12.4.24 setRotationSpeed()	91
5.12.4.25 startMoving()	91
5.12.4.26 startRotating()	91
5.12.4.27 stopMoving()	91
5.12.4.28 stopRotating()	91
5.12.4.29 toggleActive()	92
5.12.4.30 toJSON()	92
5.12.4.31 type()	92
5.12.4.32 willCollide()	92
5.12.5 Member Data Documentation	93
5.12.5.1 active	93
5.12.5.2 isMoving	93
5.12.5.3 isRotating	93
5.12.5.4 move_speed	93
5.12.5.5 rotation_speed	93
5.12.5.6 timeConstant	93
5.13 SimulationEngine Class Reference	94
5.13.1 Constructor & Destructor Documentation	96
5.13.1.1 SimulationEngine()	96
5.13.1.2 ~SimulationEngine()	96
5.13.2 Member Function Documentation	97
5.13.2.1 clearScene()	97
5.13.2.2 getControlledRobot()	97
5.13.2.3 getFPS()	97
5.13.2.4 getFrameTime()	97
5.13.2.5 getSimulationSpeed()	97
5.13.2.6 getTimeConstant()	98
5.13.2.7 isInsideScene()	98
5.13.2.8 loadSimulation()	98
5.13.2.9 read()	98
5.13.2.10 saveSimulation()	99
5.13.2.11 setControlledRobot()	99
5.13.2.12 setFPS()	99
5.13.2.13 setSimulationSpeed()	100
5.13.2.14 toJson()	100
5.13.2.15 updateTimeConstant()	100
5.13.3 Member Data Documentation	100
5.13.3.1 controlledRobot	100
5.13.3.2 fps	101

5.13.3.3 <code>simulationSpeed</code>	101
5.13.3.4 <code>timeConstant</code>	101

Index	103
--------------	------------

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Ui	7
--------------------------	-------------------

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

GameObject	33
Obstacle	48
Robot	80
AutoRobot	9
QDialog	
PopupSaveWindow	77
QGraphicsEllipseItem	
Robot	80
QGraphicsRectItem	
Obstacle	48
QGraphicsScene	
SimulationEngine	94
QLineEdit	
ParamEditLine	62
QMainWindow	
MainWindow	38
QObject	
Obstacle	48
Robot	80
QPushButton	
CheckableButton	20
ExpButton	30
QWidget	
ExpandableButtonWidget	25
OverlayWidget	55
ParamWidget	65

Chapter 3

Class Index

3.1 Class List

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

AutoRobot	A class to represent an autonomous robot	9
CheckableButton	A class to represent a checkable button	20
ExpandableButtonWidget	A class to represent an expandable button widget	25
ExpButton	A class for expandable buttons	30
GameObject	A class to represent a game object in the simulation	33
MainWindow	A class to represent the main window of the application	38
Obstacle	A class to represent an obstacle	48
OverlayWidget	A class to represent an overlay widget	55
ParamEditLine	A class to represent a line edit widget for editing parameters	62
ParamWidget	A class to represent a widget for editing parameters of game objects	65
PopupSaveWindow	A class to represent a popup save window	77
Robot	A class to represent a robot in the simulation. By default, the robot is a circle with a line drawn to represent its direction	80
SimulationEngine	94

Chapter 4

Namespace Documentation

4.1 Ui Namespace Reference

Chapter 5

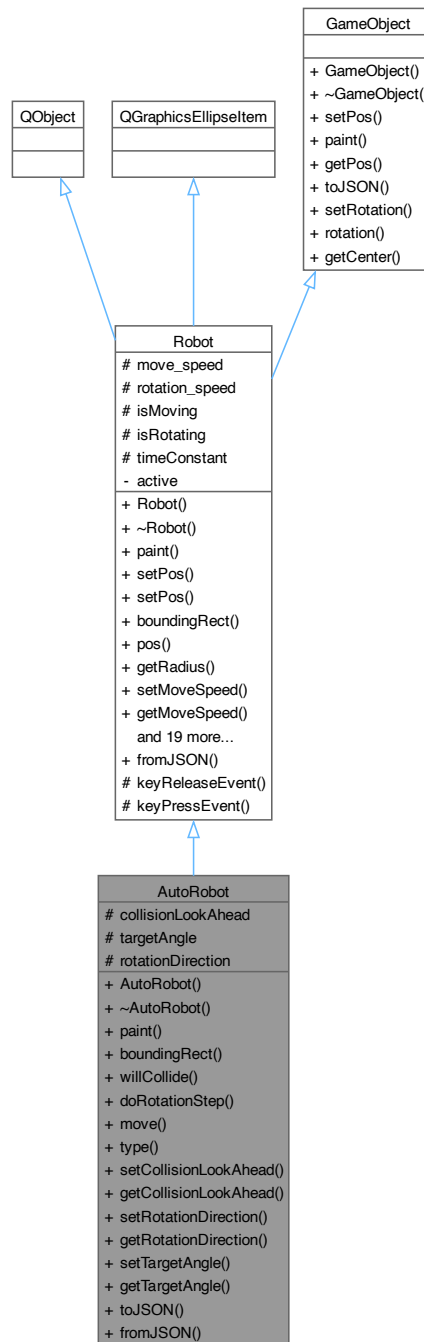
Class Documentation

5.1 AutoRobot Class Reference

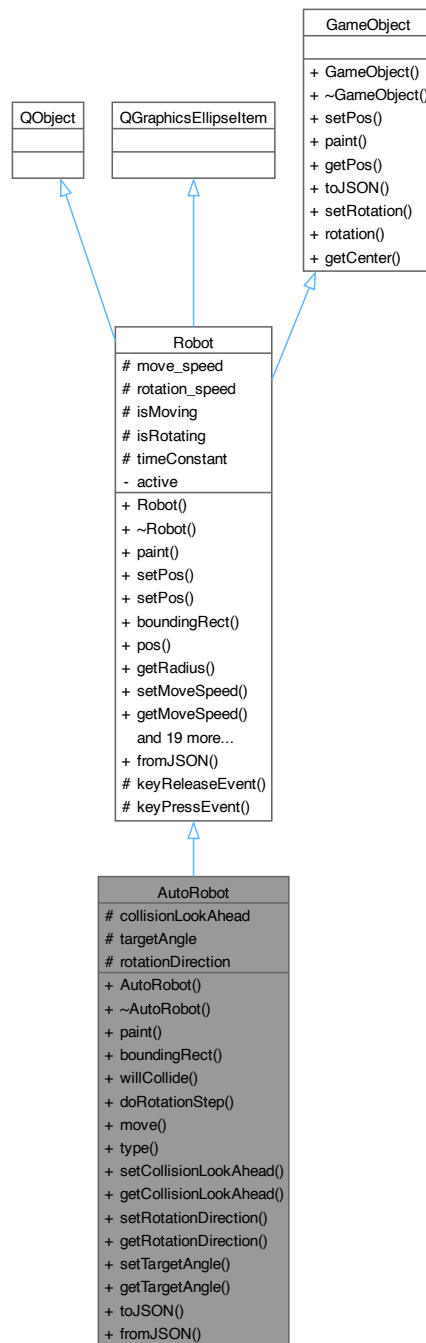
A class to represent an autonomous robot.

```
#include <autorobot.hpp>
```

Inheritance diagram for AutoRobot:



Collaboration diagram for AutoRobot:



Public Types

- enum { `Type` = `QGraphicsItem::UserType + 2` }

Public Types inherited from **Robot**

- enum `RotationDirection` { `Left` = -1 , `None` = 0 , `Right` = 1 }

Enum to represent the direction of rotation of the robot.

- enum { `Type` = QGraphicsItem::UserType + 1 }

Public Member Functions

- `AutoRobot` (QGraphicsItem *parent=nullptr, qreal size=50, qreal `collisionLookAhead`=10, `Robot::RotationDirection` `rotationDirection`=`Robot::RotationDirection::Right`, qreal moveSpeed=1, qreal rotationSpeed=1, qreal *timeConstant=nullptr)

Constructor for `AutoRobot`.

- `~AutoRobot` ()
- void `paint` (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override
- QRectF `boundingRect` () const override
- bool `willCollide` (QPointF directionVector, qreal magnitude, bool allowAnticollision) override
- void `doRotationStep` (`RotationDirection` direction)
- bool `move` () override
- int `type` () const override
- void `setCollisionLookAhead` (qreal lookAhead)
- qreal `getCollisionLookAhead` ()
- void `setRotationDirection` (`RotationDirection` direction)
- `RotationDirection` `getRotationDirection` ()
- void `setTargetAngle` (qreal angle)
- qreal `getTargetAngle` ()
- QJsonObject `toJSON` () override

Public Member Functions inherited from `Robot`

- `Robot` (QGraphicsItem *parent=nullptr, qreal *timeConstant=nullptr)
- `~Robot` ()
- void `setPos` (const QPointF &pos)
- void `setPos` (qreal x, qreal y) override
- QPointF `pos` ()
- qreal `getRadius` () const
- void `setMoveSpeed` (qreal speed)
- qreal `getMoveSpeed` ()
- void `setRotationSpeed` (qreal speed)

- qreal [getRotationSpeed](#) ()
Get the rotation speed of the robot.
- void [startMoving](#) ()
Allow the robot to be moved by setting the isMoving flag to true.
- void [stopMoving](#) ()
Stop the robot from moving by setting the isMoving flag to false.
- void [startRotating](#) ([RotationDirection](#) direction)
Start rotating the robot in the given direction.
- void [stopRotating](#) ()
Stop the robot from rotating by setting the isRotating flag to None.
- QPointF [getDirectionVector](#) ()
Get the direction vector of the robot.
- int [type](#) () const override
Get the type of the robot.
- QPointF [getPos](#) () override
Get the position of the robot.
- void [toggleActive](#) ()
Toggle the active state of the robot.
- bool [isActive](#) ()
Check if the robot is active.
- qreal [getAngle](#) ()
Get the angle of the robot.
- void [setRadius](#) (qreal radius)
Set the angle of the robot.
- QPointF [getCenter](#) () override
Get the center of the robot.
- qreal [rotation](#) () override
Get the time constant of the simulation.
- void [setRotation](#) (qreal angle) override
Set the rotation of the robot.

Public Member Functions inherited from [GameObject](#)

- [GameObject](#) ()=default
- [~GameObject](#) ()=default

Static Public Member Functions

- static [AutoRobot](#) * [fromJSON](#) (const QJsonObject &object, qreal *[timeConstant](#))
Create an [AutoRobot](#) object from a JSON object.

Static Public Member Functions inherited from [Robot](#)

- static [Robot](#) * [fromJSON](#) (const QJsonObject &object, qreal *[timeConstant](#))
Create a [Robot](#) object from a JSON object.

Protected Attributes

- qreal `collisionLookAhead` = 0
The look ahead distance for collision detection.
- qreal `targetAngle` = 0
The target angle of the robot.
- `Robot::RotationDirection` `rotationDirection` = `Robot::RotationDirection::Right`
The rotation direction of the robot.

Protected Attributes inherited from `Robot`

- qreal `move_speed` = 1
The speed of the robot.
- qreal `rotation_speed` = 1
The speed of the rotation of the robot.
- bool `isMoving` = false
Flag to indicate if the robot is moving.
- `RotationDirection` `isRotating` = `RotationDirection::None`
Flag to indicate the direction of rotation.
- qreal * `timeConstant` = nullptr
The time constant of the simulation.

Additional Inherited Members

Signals inherited from `Robot`

- void `paramsUpdated` ()
Signal emitted when the parameters of the robot are updated.
- void `robotSepuku` ()
Signal emitted when the robot is removed.

Protected Member Functions inherited from `Robot`

- void `keyReleaseEvent` (QKeyEvent *event)
The radius of the robot.
- void `keyPressEvent` (QKeyEvent *event)
Overridden keyPressEvent method.

5.1.1 Detailed Description

A class to represent an autonomous robot.

This class inherits from `Robot` and provides functionalities for an autonomous robot.

See also

[Robot](#)

5.1.2 Member Enumeration Documentation

5.1.2.1 anonymous enum

anonymous enum

Enumerator

Type	
------	--

5.1.3 Constructor & Destructor Documentation

5.1.3.1 AutoRobot()

```
AutoRobot::AutoRobot (
    QGraphicsItem * parent = nullptr,
    qreal size = 50,
    qreal collisionLookAhead = 10,
    Robot::RotationDirection rotationDirection = Robot::RotationDirection::Right,
    qreal moveSpeed = 1,
    qreal rotationSpeed = 1,
    qreal * timeConstant = nullptr )
```

Constructor for [AutoRobot](#).

Parameters

<i>parent</i>	The parent QGraphicsItem.
<i>size</i>	The size of the robot.
<i>collisionLookAhead</i>	The distance the robot looks ahead for collisions.
<i>rotationDirection</i>	The initial rotation direction of the robot.
<i>moveSpeed</i>	The movement speed of the robot.
<i>rotationSpeed</i>	The rotation speed of the robot.
<i>timeConstant</i>	A pointer to the time constant.

5.1.3.2 ~AutoRobot()

```
AutoRobot::~AutoRobot ( )
```

5.1.4 Member Function Documentation

5.1.4.1 boundingRect()

```
QRectF AutoRobot::boundingRect ( ) const [override], [virtual]
```

Reimplemented from [Robot](#).

5.1.4.2 doRotationStep()

```
void AutoRobot::doRotationStep (
    RotationDirection direction )
```

Perform a rotation step.

Parameters

<i>direction</i>	The direction of the rotation
------------------	-------------------------------

Returns

void

5.1.4.3 fromJSON()

```
static AutoRobot * AutoRobot::fromJSON (
    const QJsonObject & object,
    qreal * timeConstant ) [static]
```

Create an [AutoRobot](#) object from a JSON object.

Parameters

<i>object</i>	The JSON object to create the AutoRobot object from
<i>timeConstant</i>	The time constant of the robot

Returns

[AutoRobot](#)* The [AutoRobot](#) object created from the JSON object

5.1.4.4 getCollisionLookAhead()

```
qreal AutoRobot::getCollisionLookAhead ( ) [inline]
```

Get the look ahead distance for collision detection.

Returns

qreal The look ahead distance

5.1.4.5 getRotationDirection()

```
RotationDirection AutoRobot::getRotationDirection ( ) [inline]
```

Get the rotation direction of the robot.

Returns

[RotationDirection](#) The rotation direction

5.1.4.6 getTargetAngle()

```
qreal AutoRobot::getTargetAngle ( ) [inline]
```

Get the target angle of the robot.

Returns

qreal The target angle

5.1.4.7 move()

```
bool AutoRobot::move ( ) [override], [virtual]
```

Perform a movement step.

Returns

bool Whether the movement step was successful

Reimplemented from [Robot](#).

5.1.4.8 paint()

```
void AutoRobot::paint (
    QPainter * painter,
    const QStyleOptionGraphicsItem * option,
    QWidget * widget ) [override], [virtual]
```

Override the paint method to draw a line showing the direction of the robot

Reimplemented from [Robot](#).

5.1.4.9 setCollisionLookAhead()

```
void AutoRobot::setCollisionLookAhead (
    qreal lookAhead ) [inline]
```

Set the look ahead distance for collision detection.

Parameters

<i>lookAhead</i>	The look ahead distance
------------------	-------------------------

Returns

void

5.1.4.10 setRotationDirection()

```
void AutoRobot::setRotationDirection (
    RotationDirection direction ) [inline]
```

Set the rotation direction of the robot.

Parameters

<i>direction</i>	The rotation direction
------------------	------------------------

Returns

void

5.1.4.11 setTargetAngle()

```
void AutoRobot::setTargetAngle (
    qreal angle ) [inline]
```

Set the target angle of the robot.

Parameters

<i>angle</i>	The target angle
--------------	------------------

Returns

void

5.1.4.12 toJSON()

```
QJsonObject AutoRobot::toJSON ( ) [override], [virtual]
```

Get the JSON representation of the object.

Returns

QJsonObject The JSON representation of the object

Reimplemented from [Robot](#).

5.1.4.13 type()

```
int AutoRobot::type ( ) const [inline], [override]
```

Get the type of the object.

Returns

int The type of the object

5.1.4.14 willCollide()

```
bool AutoRobot::willCollide (
    QPointF directionVector,
    qreal magnitude,
    bool allowAnticollision ) [override], [virtual]
```

Check if the robot will collide with any object in the scene.

Parameters

<i>directionVector</i>	The direction vector of the robot
<i>magnitude</i>	The magnitude of the direction vector
<i>allowAnticollision</i>	Whether to allow anticollision

Returns

bool Whether the robot will collide with any object in the scene

Reimplemented from [Robot](#).

5.1.5 Member Data Documentation

5.1.5.1 collisionLookAhead

```
qreal AutoRobot::collisionLookAhead = 0 [protected]
```

The look ahead distance for collision detection.

5.1.5.2 rotationDirection

```
Robot::RotationDirection AutoRobot::rotationDirection = Robot::RotationDirection::Right [protected]
```

The rotation direction of the robot.

5.1.5.3 targetAngle

```
qreal AutoRobot::targetAngle = 0 [protected]
```

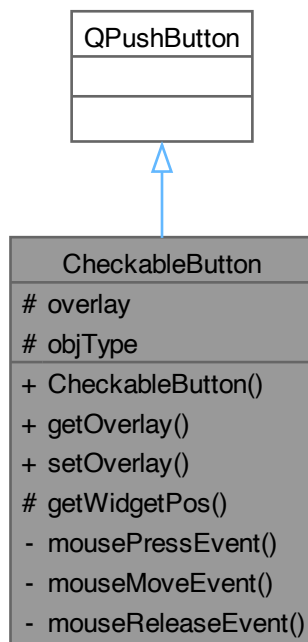
The target angle of the robot.

5.2 CheckableButton Class Reference

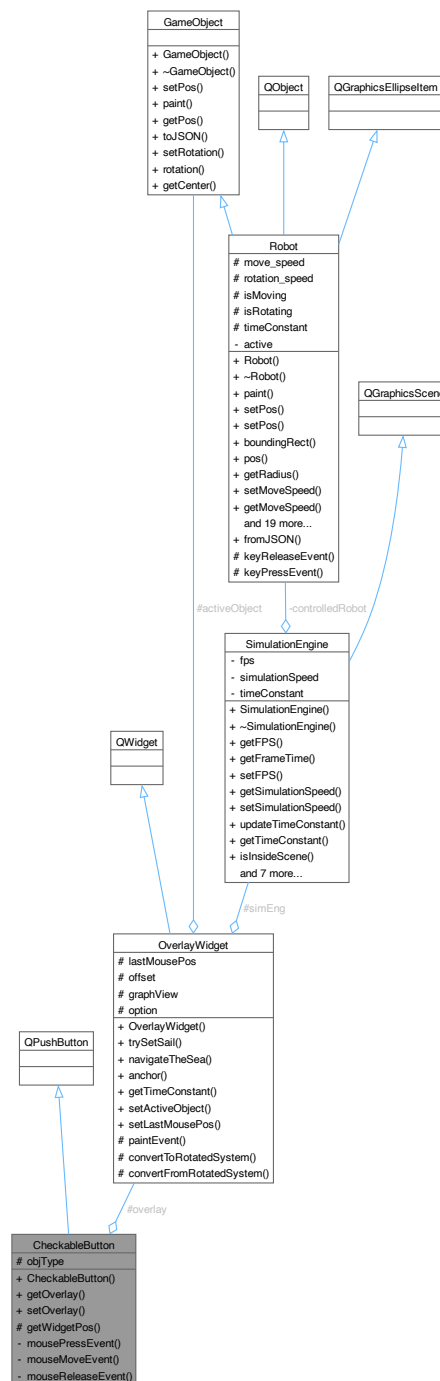
A class to represent a checkable button.

```
#include <checkablebutton.hpp>
```

Inheritance diagram for CheckableButton:



Collaboration diagram for CheckableButton:



Public Types

- enum `ObjectType` { `AUTO` , `CONT` , `OBST` }

Enum to represent the type of object that the button represents `AUTO`: *AutoRobot* `CONT`: *ControlledRobot* `OBST`: *Obstacle*.

Public Member Functions

- [CheckableButton](#) (const QString &text, QWidget *parent=nullptr, [ObjectType](#) type=[ObjectType::OBST](#))
Constructor for [CheckableButton](#).
- [OverlayWidget](#) * [getOverlay](#) () const
Get the overlay widget of the button.
- void [setOverlay](#) ([OverlayWidget](#) *overlay)
Set the overlay widget of the button.

Protected Member Functions

- QPoint [getWidgetPos](#) (QPoint localPos)
Get the position of the widget on the grid.

Protected Attributes

- [OverlayWidget](#) * overlay
Pointer to the overlay widget.
- [ObjectType](#) objType
The type of object that the button represents.

Private Slots

- void [mousePressEvent](#) (QMouseEvent *event) override
Override the [mousePressEvent](#) method.
- void [mouseMoveEvent](#) (QMouseEvent *event) override
Override the [mouseMoveEvent](#) method.
- void [mouseReleaseEvent](#) (QMouseEvent *event) override
Override the [mouseReleaseEvent](#) method.

5.2.1 Detailed Description

A class to represent a checkable button.

This class inherits from [QPushButton](#) and provides functionalities for a button that can be checked and unchecked. It also has an [OverlayWidget](#) that is used to draw the object on the grid.

See also

[QPushButton](#)

5.2.2 Member Enumeration Documentation

5.2.2.1 ObjectType

enum [CheckableButton::ObjectType](#)

Enum to represent the type of object that the button represents AUTO: [AutoRobot](#) CONT: [ControlledRobot](#) OBST: [Obstacle](#).

•

Enumerator

AUTO	
CONT	
OBST	

5.2.3 Constructor & Destructor Documentation

5.2.3.1 CheckableButton()

```
CheckableButton::CheckableButton (
    const QString & text,
    QWidget * parent = nullptr,
    ObjectType type = ObjectType::OBST ) [explicit]
```

Constructor for [CheckableButton](#).

Parameters

<i>text</i>	The text to be displayed on the button.
<i>parent</i>	The parent QWidget.
<i>type</i>	The type of object that the button represents.

5.2.4 Member Function Documentation

5.2.4.1 getOverlay()

```
OverlayWidget * CheckableButton::getOverlay ( ) const [inline]
```

Get the overlay widget of the button.

Returns

OverlayWidget* The overlay widget of the button

5.2.4.2 getWidgetPos()

```
QPoint CheckableButton::getWidgetPos (
    QPoint localPos ) [protected]
```

Get the position of the widget on the grid.

Parameters

<i>localPos</i>	The local position of the mouse.
-----------------	----------------------------------

Returns

QPoint The position in the overlay widget.

5.2.4.3 mouseMoveEvent

```
void CheckableButton::mouseMoveEvent (  
    QMouseEvent * event ) [override], [private], [slot]
```

Override the mouseMoveEvent method.

Parameters

<i>event</i>	The mouse event
--------------	-----------------

Returns

void

5.2.4.4 mousePressEvent

```
void CheckableButton::mousePressEvent (  
    QMouseEvent * event ) [override], [private], [slot]
```

Override the mousePressEvent method.

Parameters

<i>event</i>	The mouse event
--------------	-----------------

Returns

void

5.2.4.5 mouseReleaseEvent

```
void CheckableButton::mouseReleaseEvent (  
    QMouseEvent * event ) [override], [private], [slot]
```

Override the mouseReleaseEvent method.

Parameters

<i>event</i>	The mouse event
--------------	-----------------

Returns

void

5.2.4.6 setOverlay()

```
void CheckableButton::setOverlay (
    OverlayWidget * overlay ) [inline]
```

Set the overlay widget of the button.

Parameters

<i>overlay</i>	The overlay widget to set
----------------	---------------------------

Returns

void

5.2.5 Member Data Documentation**5.2.5.1 objType**

```
ObjectType CheckableButton::objType [protected]
```

The type of object that the button represents.

5.2.5.2 overlay

```
OverlayWidget* CheckableButton::overlay [protected]
```

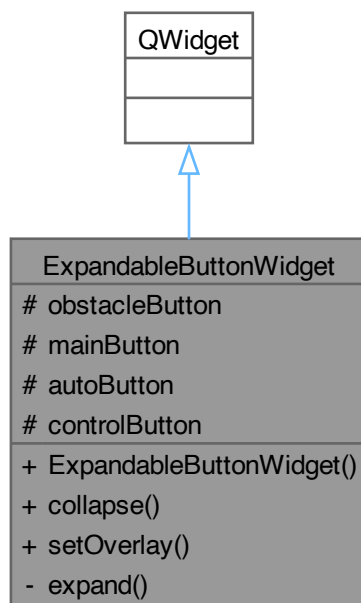
Pointer to the overlay widget.

5.3 ExpandableButtonWidget Class Reference

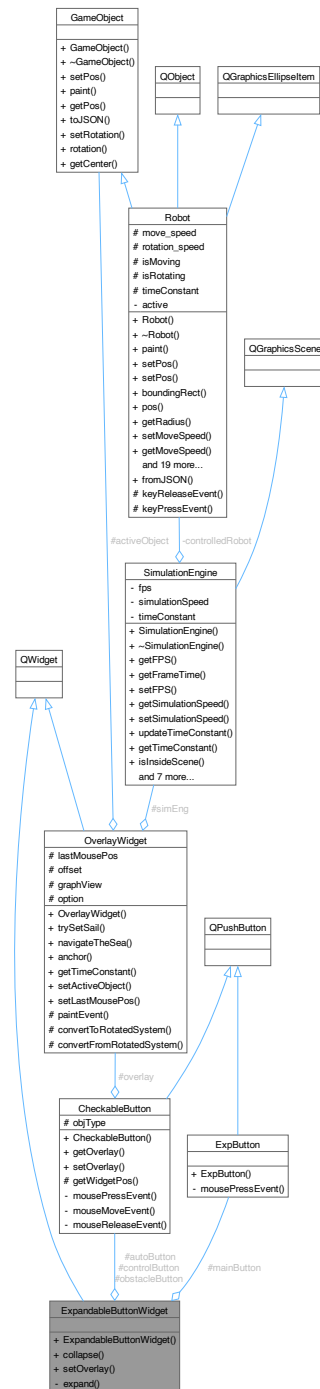
A class to represent an expandable button widget.

```
#include <expbuttonwidget.hpp>
```

Inheritance diagram for ExpandableButtonWidget:



Collaboration diagram for ExpandableButtonWidget:



Public Member Functions

- `ExpandableButtonWidget` (`QWidget *parent=nullptr`)
Construct a new Expandable Button Widget object.
- `void collapse ()`
Get the obstacle button.
- `void setOverlay (OverlayWidget *overlay)`
Get the obstacle button.

Protected Attributes

- [CheckableButton](#) * `obstacleButton`
Reference to the obstacle button.
- [ExpButton](#) * `mainButton`
Reference to the main button.
- [CheckableButton](#) * `autoButton`
Reference to the auto button.
- [CheckableButton](#) * `controlButton`
Reference to the control button.

Private Slots

- void [expand](#) ()
Slot to handle the main button press event.

5.3.1 Detailed Description

A class to represent an expandable button widget.

This class provides an interface for creating and managing expandable button widgets.

See also

[QWidget](#)

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ExpandableButtonWidget()

```
ExpandableButtonWidget::ExpandableButtonWidget (
    QWidget * parent = nullptr ) [explicit]
```

Construct a new Expandable Button Widget object.

Parameters

<i>parent</i>	The parent widget. Default is nullptr.
---------------	--

5.3.3 Member Function Documentation

5.3.3.1 collapse()

```
void ExpandableButtonWidget::collapse ( )
```

Get the obstacle button.

Returns

CheckableButton* The obstacle button.

5.3.3.2 expand

```
void ExpandableButtonWidget::expand ( ) [private], [slot]
```

Slot to handle the main button press event.

Returns

void

5.3.3.3 setOverlay()

```
void ExpandableButtonWidget::setOverlay (
    OverlayWidget * overlay )
```

Get the obstacle button.

Returns

CheckableButton* The obstacle button.

5.3.4 Member Data Documentation**5.3.4.1 autoButton**

```
CheckableButton* ExpandableButtonWidget::autoButton [protected]
```

Reference to the auto button.

5.3.4.2 controlButton

```
CheckableButton* ExpandableButtonWidget::controlButton [protected]
```

Reference to the control button.

5.3.4.3 mainButton

```
ExpButton* ExpandableButtonWidget::mainButton [protected]
```

Reference to the main button.

5.3.4.4 obstacleButton

`CheckableButton* ExpandableButtonWidget::obstacleButton` [protected]

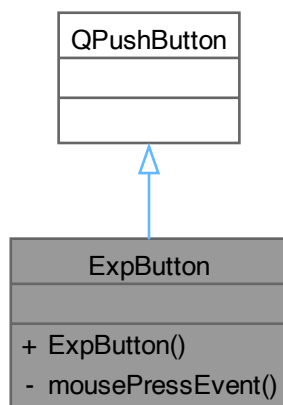
Reference to the obstacle button.

5.4 ExpButton Class Reference

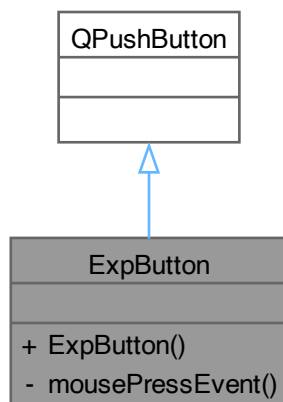
A class for expandable buttons.

```
#include <expbutton.hpp>
```

Inheritance diagram for ExpButton:



Collaboration diagram for ExpButton:



Signals

- void [pressed](#) ()
Signal emitted when the button is pressed.

Public Member Functions

- [ExpButton](#) (const QString &text, QWidget *parent=nullptr)
Constructor for [ExpButton](#).

Private Slots

- void [mousePressEvent](#) (QMouseEvent *event) override
Slot to handle the button press event.

5.4.1 Detailed Description

A class for expandable buttons.

This class inherits from QPushButton and emits a signal when pressed.

See also

QPushButton

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ExpButton()

```
ExpButton::ExpButton (  
    const QString & text,  
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [ExpButton](#).

Parameters

<i>text</i>	The text to be displayed on the button.
<i>parent</i>	The parent QWidget.

5.4.3 Member Function Documentation

5.4.3.1 mousePressEvent

```
void ExpButton::mousePressEvent (  
    QMouseEvent * event ) [override], [private], [slot]
```

Slot to handle the button press event.

Parameters

<i>event</i>	The QMouseEvent that triggered the slot.
--------------	--

Returns

void

5.4.3.2 pressed

```
void ExpButton::pressed ( ) [signal]
```

Signal emitted when the button is pressed.

Returns

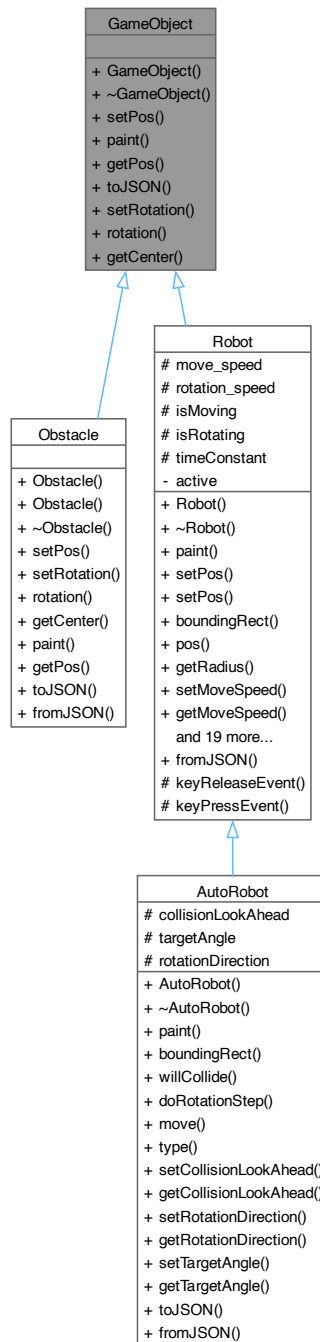
void

5.5 GameObject Class Reference

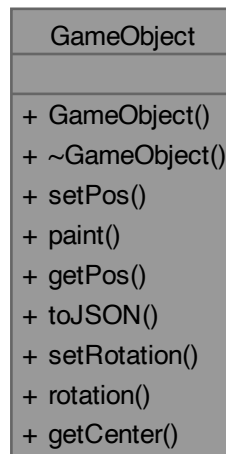
A class to represent a game object in the simulation.

```
#include <gameobject.hpp>
```

Inheritance diagram for GameObject:



Collaboration diagram for GameObject:



Public Member Functions

- [GameObject](#) ()=default
- [~GameObject](#) ()=default
- virtual void [setPos](#) (qreal x, qreal y)=0
Set the position of the game object.
- virtual void [paint](#) (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)=0
Paint the game object.
- virtual QPointF [getPos](#) ()=0
Get the position of the game object.
- virtual QJsonObject [toJSON](#) ()=0
Convert the game object to a JSON object.
- virtual void [setRotation](#) (qreal angle)=0
Set the rotation of the game object.
- virtual qreal [rotation](#) ()=0
Get the rotation of the game object.
- virtual QPointF [getCenter](#) ()=0
Get the center of the game object.

5.5.1 Detailed Description

A class to represent a game object in the simulation.

This class provides an interface for creating and managing game objects.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 GameObject()

```
GameObject::GameObject ( ) [default]
```

5.5.2.2 ~GameObject()

```
GameObject::~~GameObject ( ) [default]
```

5.5.3 Member Function Documentation

5.5.3.1 getCenter()

```
virtual QPointF GameObject::getCenter ( ) [pure virtual]
```

Get the center of the game object.

Returns

QPointF

Implemented in [Obstacle](#), and [Robot](#).

5.5.3.2 getPos()

```
virtual QPointF GameObject::getPos ( ) [pure virtual]
```

Get the position of the game object.

Returns

QPointF

Implemented in [Obstacle](#), and [Robot](#).

5.5.3.3 paint()

```
virtual void GameObject::paint (
    QPainter * painter,
    const QStyleOptionGraphicsItem * option,
    QWidget * widget ) [pure virtual]
```

Paint the game object.

Parameters

<i>painter</i>	
<i>option</i>	
<i>widget</i>	

Returns

void

Implemented in [AutoRobot](#), [Obstacle](#), and [Robot](#).

5.5.3.4 rotation()

```
virtual qreal GameObject::rotation ( ) [pure virtual]
```

Get the rotation of the game object.

Returns

qreal

Implemented in [Obstacle](#), and [Robot](#).

5.5.3.5 setPos()

```
virtual void GameObject::setPos (
    qreal x,
    qreal y ) [pure virtual]
```

Set the position of the game object.

Parameters

<i>x</i>	
<i>y</i>	

Returns

void

Implemented in [Obstacle](#), and [Robot](#).

5.5.3.6 setRotation()

```
virtual void GameObject::setRotation (
    qreal angle ) [pure virtual]
```

Set the rotation of the game object.

Parameters

<i>angle</i>	
--------------	--

Returns

void

Implemented in [Obstacle](#), and [Robot](#).**5.5.3.7 toJSON()**

```
virtual QJsonObject GameObject::toJSON ( ) [pure virtual]
```

Convert the game object to a JSON object.

Returns

QJsonObject

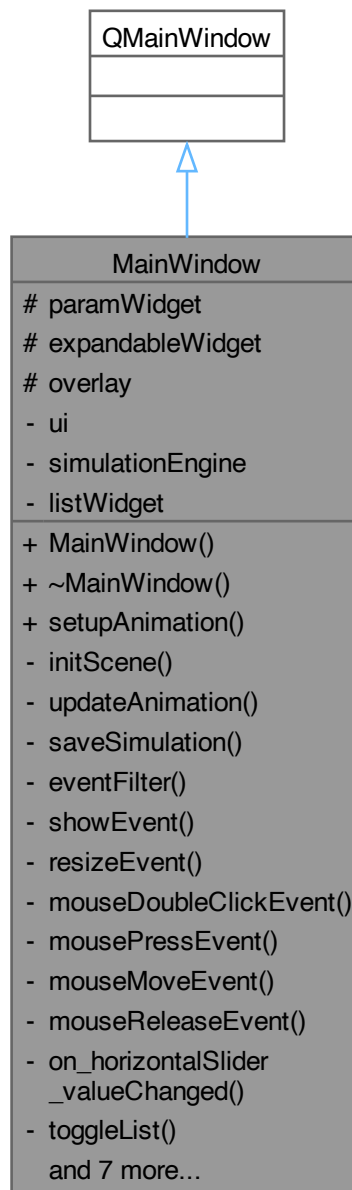
Implemented in [AutoRobot](#), [Obstacle](#), and [Robot](#).

5.6 MainWindow Class Reference

A class to represent the main window of the application.

```
#include <mainwindow.h>
```


Inheritance diagram for MainWindow:



Protected Attributes

- [ParamWidget](#) * [paramWidget](#)
The param widget.
- [ExpandableButtonWidget](#) * [expandableWidget](#)
The expandable button widget.
- [OverlayWidget](#) * [overlay](#)
The overlay widget.

Private Slots

- void [saveSimulation](#) ()
Slot to handle the save button click event.
- bool [eventFilter](#) (QObject *object, QEvent *event) override
Overridden event filter method to handle key press events.
- void [showEvent](#) (QShowEvent *event) override
Overridden show event method to handle the show event.
- void [resizeEvent](#) (QResizeEvent *event) override
Overridden resize event method to handle the resize event.
- void [mouseDoubleClickEvent](#) (QMouseEvent *event) override
Overridden close event method to handle the close event.
- void [mousePressEvent](#) (QMouseEvent *event) override
Overridden mouse press event method to handle the mouse press event.
- void [mouseMoveEvent](#) (QMouseEvent *event) override
Overridden mouse move event method to handle the mouse move event.
- void [mouseReleaseEvent](#) (QMouseEvent *event) override
Overridden mouse release event method to handle the mouse release event.
- void [on_horizontalSlider_valueChanged](#) (int value)
Slot to handle the horizontal slider value changed event.
- void [toggleList](#) ()
Slot to handle toggling the list.
- void [handleItemDoubleClick](#) (QListWidgetItem *item)
Slot to handle the item double click event from the list.
- void [on_pushButton_clicked](#) ()
Slot to handle clear button click event.
- void [goLeft](#) ()
Slot to handle rotate anticlockwise button click event.
- void [stopRotating](#) ()
Slot to handle stop rotating button click event.
- void [goRight](#) ()
Slot to handle rotate clockwise button click event.
- void [goStraight](#) ()
Slot to handle move forward button click event.
- void [stopMoving](#) ()
Slot to handle stop moving button click event.

Private Member Functions

- void [initScene](#) ()
- void [updateAnimation](#) ()

Private Attributes

- `Ui::MainWindow * ui`
The UI object.
- `SimulationEngine * simulationEngine`
The simulation engine.
- `QListWidget * listWidget`
The list widget.

5.6.1 Detailed Description

A class to represent the main window of the application.

This class inherits from QMainWindow and provides the main window of the application.

See also

QMainWindow

5.6.2 Constructor & Destructor Documentation

5.6.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr )
```

5.6.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

5.6.3 Member Function Documentation

5.6.3.1 eventFilter

```
bool MainWindow::eventFilter (
    QObject * object,
    QEvent * event ) [override], [private], [slot]
```

Overridden event filter method to handle key press events.

Parameters

<i>object</i>	The object that the event is being filtered for
<i>event</i>	The event that is being filtered

Returns

bool Whether the event was handled

5.6.3.2 goLeft

```
void MainWindow::goLeft ( ) [private], [slot]
```

Slot to handle rotate anticlockwise button click event.

Returns

void

5.6.3.3 goRight

```
void MainWindow::goRight ( ) [private], [slot]
```

Slot to handle rotate clockwise button click event.

Returns

void

5.6.3.4 goStraight

```
void MainWindow::goStraight ( ) [private], [slot]
```

Slot to handle move forward button click event.

Returns

void

5.6.3.5 handleItemDoubleClick

```
void MainWindow::handleItemDoubleClick (
    QListWidgetItem * item ) [private], [slot]
```

Slot to handle the item double click event from the list.

Parameters

<i>item</i>	The item that was double clicked
-------------	----------------------------------

Returns

void

5.6.3.6 initScene()

```
void MainWindow::initScene ( ) [private]
```

5.6.3.7 mouseDoubleClickEvent

```
void MainWindow::mouseDoubleClickEvent (
    QMouseEvent * event ) [override], [private], [slot]
```

Overriden close event method to handle the close event.

Parameters

<i>event</i>	The close event
--------------	-----------------

Returns

void

5.6.3.8 mouseMoveEvent

```
void MainWindow::mouseMoveEvent (
    QMouseEvent * event ) [override], [private], [slot]
```

Overriden mouse move event method to handle the mouse move event.

Parameters

<i>event</i>	The mouse move event
--------------	----------------------

Returns

void

5.6.3.9 mousePressEvent

```
void MainWindow::mousePressEvent (
    QMouseEvent * event ) [override], [private], [slot]
```

Overriden mouse press event method to handle the mouse press event.

Parameters

<i>event</i>	The mouse press event
--------------	-----------------------

Returns

void

5.6.3.10 mouseReleaseEvent

```
void MainWindow::mouseReleaseEvent (  
    QMouseEvent * event ) [override], [private], [slot]
```

Overriden mouse release event method to handle the mouse release event.

Parameters

<i>event</i>	The mouse release event
--------------	-------------------------

Returns

void

5.6.3.11 on_horizontalSlider_valueChanged

```
void MainWindow::on_horizontalSlider_valueChanged (  
    int value ) [private], [slot]
```

Slot to handle the horizontal slider value changed event.

Parameters

<i>value</i>	The new value of the slider
--------------	-----------------------------

Returns

void

5.6.3.12 on_pushButton_clicked

```
void MainWindow::on_pushButton_clicked ( ) [private], [slot]
```

Slot to handle clear button click event.

Returns

void

5.6.3.13 `resizeEvent`

```
void MainWindow::resizeEvent (
    QResizeEvent * event ) [override], [private], [slot]
```

Overriden resize event method to handle the resize event.

Parameters

<i>event</i>	The resize event
--------------	------------------

Returns

void

5.6.3.14 `saveSimulation`

```
void MainWindow::saveSimulation ( ) [private], [slot]
```

Slot to handle the save button click event.

Returns

void

5.6.3.15 `setupAnimation()`

```
void MainWindow::setupAnimation ( )
```

5.6.3.16 `showEvent`

```
void MainWindow::showEvent (
    QShowEvent * event ) [override], [private], [slot]
```

Overriden show event method to handle the show event.

Parameters

<i>event</i>	The show event
--------------	----------------

Returns

void

5.6.3.17 `stopMoving`

```
void MainWindow::stopMoving ( ) [private], [slot]
```


Slot to handle stop moving button click event.

Returns

void

5.6.3.18 stopRotating

```
void MainWindow::stopRotating ( ) [private], [slot]
```

Slot to handle stop rotating button click event.

Returns

void

5.6.3.19 toggleList

```
void MainWindow::toggleList ( ) [private], [slot]
```

Slot to handle toggling the list.

Returns

void

5.6.3.20 updateAnimation()

```
void MainWindow::updateAnimation ( ) [private]
```

5.6.4 Member Data Documentation

5.6.4.1 expandableWidget

```
ExpandableButtonWidget* MainWindow::expandableWidget [protected]
```

The expandable button widget.

5.6.4.2 listWidget

```
QListWidget* MainWindow::listWidget [private]
```

The list widget.

5.6.4.3 overlay

```
OverlayWidget* MainWindow::overlay [protected]
```

The overlay widget.

5.6.4.4 paramWidget

```
ParamWidget* MainWindow::paramWidget [protected]
```

The param widget.

5.6.4.5 simulationEngine

```
SimulationEngine* MainWindow::simulationEngine [private]
```

The simulation engine.

5.6.4.6 ui

```
Ui::MainWindow* MainWindow::ui [private]
```

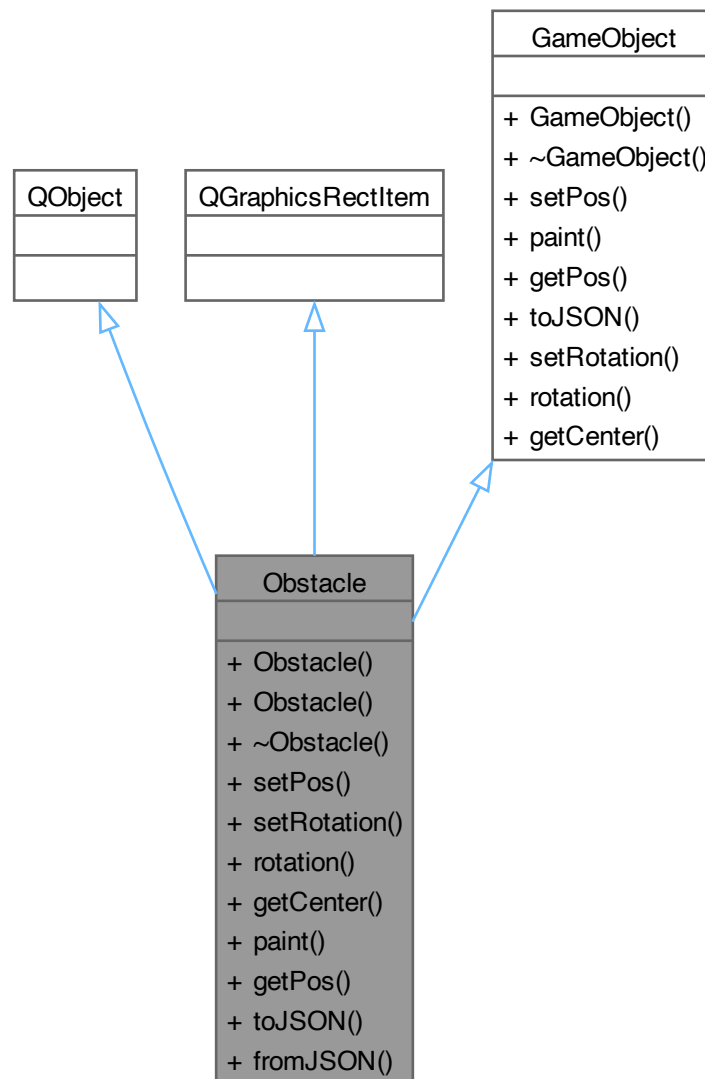
The UI object.

5.7 Obstacle Class Reference

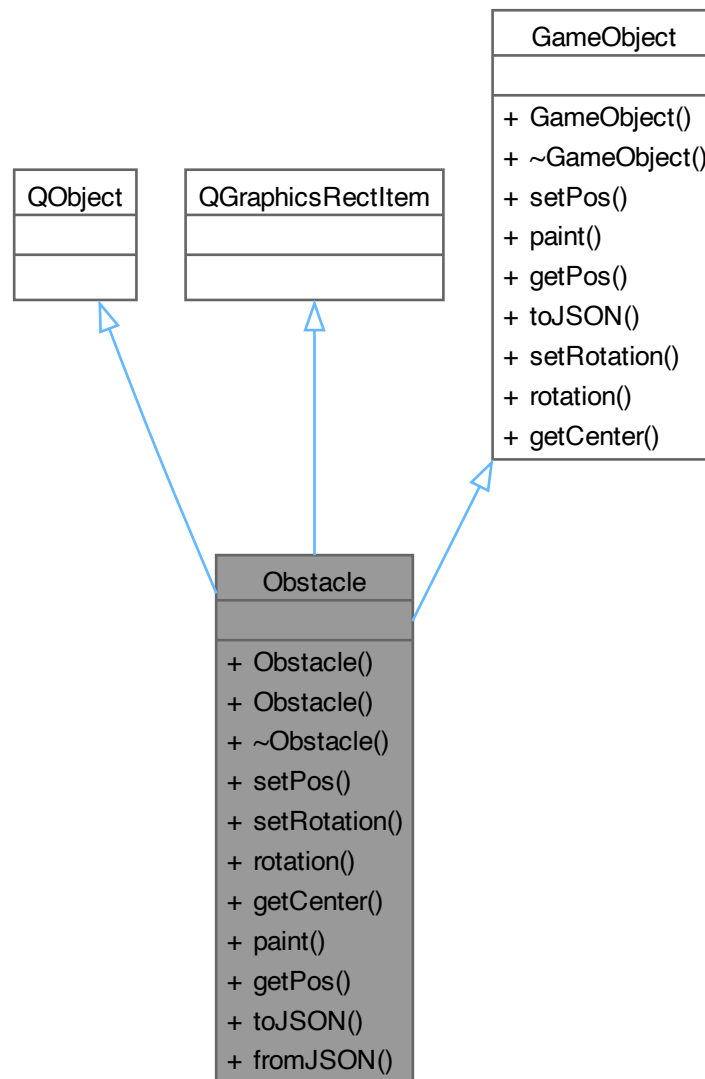
A class to represent an obstacle.

```
#include <obstacle.hpp>
```

Inheritance diagram for Obstacle:



Collaboration diagram for Obstacle:



Signals

- void [paramsUpdated](#) ()
Signal emitted when the parameters of the obstacle are updated.
- void [obstacleSepuku](#) ()
Signal emitted when the obstacle is removed.

Public Member Functions

- [Obstacle](#) (QGraphicsItem *parent=nullptr)
Default constructor.

- [Obstacle](#) (const [Obstacle](#) &)
Copy constructor.
- [~Obstacle](#) ()
Destructor.
- void [setPos](#) (qreal x, qreal y) override
Set the position of the obstacle.
- void [setRotation](#) (qreal angle) override
Set the rotation of the obstacle.
- qreal [rotation](#) () override
Get the rotation of the obstacle.
- QPointF [getCenter](#) () override
Get the center of the obstacle.
- void [paint](#) (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override
Paint the obstacle.
- QPointF [getPos](#) () override
Get the position of the obstacle.
- QJsonObject [toJSON](#) () override
Convert the obstacle to a JSON object.

Public Member Functions inherited from [GameObject](#)

- [GameObject](#) ()=default
- [~GameObject](#) ()=default

Static Public Member Functions

- static [Obstacle](#) * [fromJSON](#) (const QJsonObject &json)
Create an [Obstacle](#) object from a JSON object.

5.7.1 Detailed Description

A class to represent an obstacle.

This class inherits from QGraphicsRectItem and [GameObject](#). It represents an obstacle in a game.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Obstacle() [1/2]

```
Obstacle::Obstacle (
    QGraphicsItem * parent = nullptr )
```

Default constructor.

Parameters

<i>parent</i>	The parent QGraphicsItem.
---------------	---------------------------

Returns

void

5.7.2.2 Obstacle() [2/2]

```
Obstacle::Obstacle (
    const Obstacle & ) [inline]
```

Copy constructor.

Parameters

<i>Obstacle</i>	The <i>Obstacle</i> object to copy.
-----------------	-------------------------------------

Returns

void

5.7.2.3 ~Obstacle()

```
Obstacle::~Obstacle ( )
```

Destructor.

5.7.3 Member Function Documentation**5.7.3.1 fromJSON()**

```
static Obstacle * Obstacle::fromJSON (
    const QJsonObject & json ) [static]
```

Create an *Obstacle* object from a JSON object.

Parameters

<i>json</i>	The QJsonObject to convert.
-------------	-----------------------------

Returns

A pointer to the created *Obstacle* object.

5.7.3.2 getCenter()

```
QPointF Obstacle::getCenter ( ) [inline], [override], [virtual]
```

Get the center of the obstacle.

Returns

The center of the obstacle as a QPointF.

Implements [GameObject](#).

5.7.3.3 getPos()

```
QPointF Obstacle::getPos ( ) [override], [virtual]
```

Get the position of the obstacle.

Returns

The position of the obstacle as a QPointF object.

Implements [GameObject](#).

5.7.3.4 obstacleSepuku

```
void Obstacle::obstacleSepuku ( ) [signal]
```

Signal emitted when the obstacle is removed.

Returns

void

5.7.3.5 paint()

```
void Obstacle::paint (
    QPainter * painter,
    const QStyleOptionGraphicsItem * option,
    QWidget * widget ) [override], [virtual]
```

Paint the obstacle.

Parameters

<i>painter</i>	Pointer to the QPainter object.
<i>option</i>	Pointer to the QStyleOptionGraphicsItem object.
<i>widget</i>	Pointer to the QWidget object.

Implements [GameObject](#).

5.7.3.6 paramsUpdated

```
void Obstacle::paramsUpdated ( ) [signal]
```

Signal emitted when the parameters of the obstacle are updated.

Returns

void

5.7.3.7 rotation()

```
qreal Obstacle::rotation ( ) [inline], [override], [virtual]
```

Get the rotation of the obstacle.

Returns

The rotation of the obstacle as a qreal.

Implements [GameObject](#).

5.7.3.8 setPos()

```
void Obstacle::setPos (
    qreal x,
    qreal y ) [override], [virtual]
```

Set the position of the obstacle.

Parameters

<i>x</i>	The x-coordinate of the position.
<i>y</i>	The y-coordinate of the position.

Returns

void

Implements [GameObject](#).

5.7.3.9 setRotation()

```
void Obstacle::setRotation (
    qreal angle ) [override], [virtual]
```

Set the rotation of the obstacle.

Parameters

<i>angle</i>	The angle of the rotation.
--------------	----------------------------

Returns

void

Implements [GameObject](#).

5.7.3.10 toJSON()

```
QJsonObject Obstacle::toJSON ( ) [override], [virtual]
```

Convert the obstacle to a JSON object.

Returns

The obstacle as a QJsonObject.

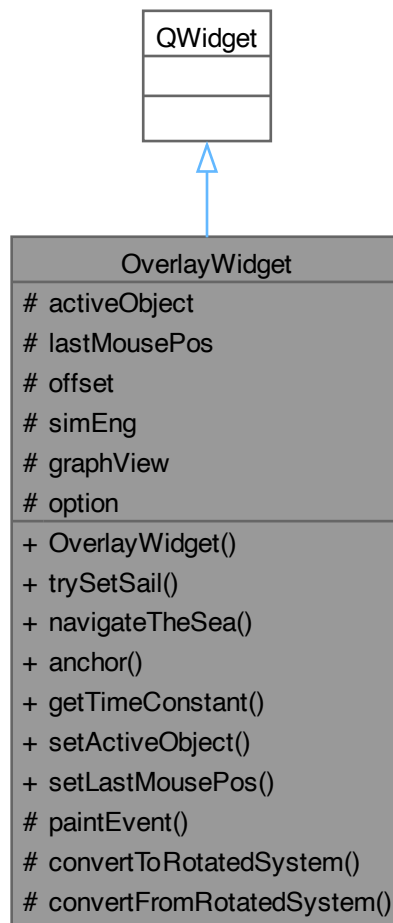
Implements [GameObject](#).

5.8 OverlayWidget Class Reference

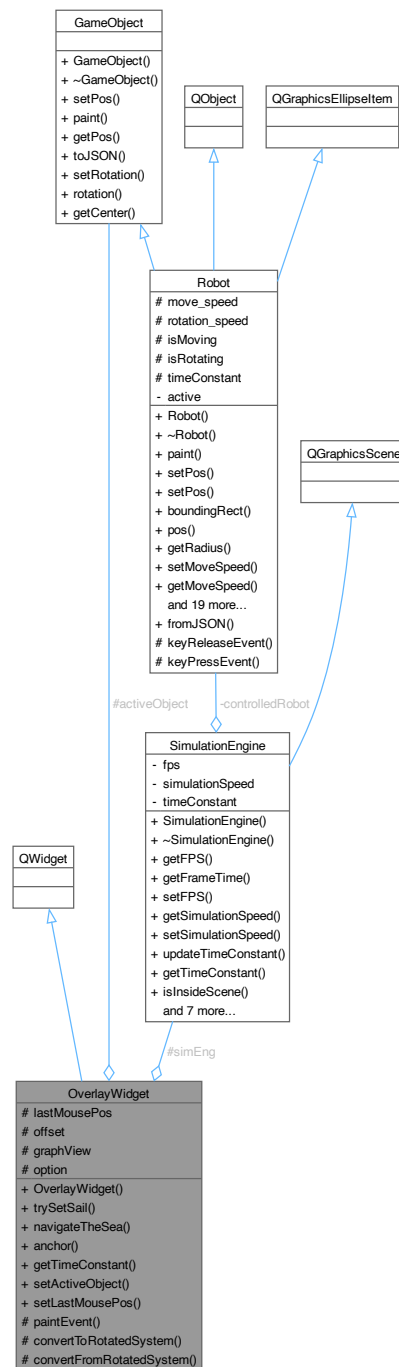
A class to represent an overlay widget.

```
#include <overlaywidget.hpp>
```

Inheritance diagram for OverlayWidget:



Collaboration diagram for OverlayWidget:



Public Member Functions

- **OverlayWidget** (**QWidget** *parent=nullptr, **SimulationEngine** *simEng=nullptr, **QGraphicsView** *graphView=nullptr)
Construct a new Overlay Widget object.
- void **trySetSail** (**QMouseEvent** *event)
Try grab the object based on the mouse position.
- void **navigateTheSea** (**QMouseEvent** *event)

- *Drag the object based on the mouse position in the overlay.*
- void [anchor](#) ()
Anchor the object based on the mouse position back to scene.
- qreal * [getTimeConstant](#) ()
Get the time constant of the simulation engine.
- void [setActiveObject](#) (GameObject *obj)
Get the active object.
- void [setLastMousePos](#) (QPoint pos)
Get the last mouse position.

Protected Member Functions

- void [paintEvent](#) (QPaintEvent *event) override
Override the mousePressEvent method.
- QPoint [convertToRotatedSystem](#) (QPoint point, qreal angle)
Convert the point to the rotated system.
- QPoint [convertFromRotatedSystem](#) (QPoint point, qreal angle)
Convert the point from the rotated system.

Protected Attributes

- [GameObject](#) * [activeObject](#)
The active object.
- QPoint [lastMousePos](#)
The last mouse position.
- QPoint [offset](#)
- [SimulationEngine](#) * [simEng](#)
The simulation engine.
- QGraphicsView * [graphView](#)
The graphics view.
- QStyleOptionGraphicsItem [option](#)
The option for the graphics item.

5.8.1 Detailed Description

A class to represent an overlay widget.

This class provides an interface for creating and managing overlay widgets.

See also

[QWidget](#)

5.8.2 Constructor & Destructor Documentation

5.8.2.1 OverlayWidget()

```
OverlayWidget::OverlayWidget (
    QWidget * parent = nullptr,
    SimulationEngine * simEng = nullptr,
    QGraphicsView * graphView = nullptr ) [explicit]
```

Construct a new Overlay Widget object.

Parameters

<i>parent</i>	The parent widget. Default is nullptr.
<i>simEng</i>	The simulation engine. Default is nullptr.
<i>graphView</i>	The graphics view. Default is nullptr.

5.8.3 Member Function Documentation

5.8.3.1 anchor()

```
void OverlayWidget::anchor ( )
```

Anchor the object based on the mouse position back to scene.

Returns

void

5.8.3.2 convertFromRotatedSystem()

```
QPoint OverlayWidget::convertFromRotatedSystem (
    QPoint point,
    qreal angle ) [protected]
```

Convert the point from the rotated system.

Parameters

<i>point</i>	The point in the rotated system.
<i>angle</i>	The angle of the rotation.

Returns

QPoint The point in the rotated system.

5.8.3.3 convertToRotatedSystem()

```
QPoint OverlayWidget::convertToRotatedSystem (
    QPoint point,
    qreal angle ) [protected]
```

Convert the point to the rotated system.

Parameters

<i>point</i>	The point in the scene.
<i>angle</i>	The angle of the rotation.

Returns

QPoint The point in the rotated system.

5.8.3.4 getTimeConstant()

```
qreal * OverlayWidget::getTimeConstant ( ) [inline]
```

Get the time constant of the simulation engine.

Returns

qreal* The time constant of the simulation engine.

5.8.3.5 navigateTheSea()

```
void OverlayWidget::navigateTheSea (
    QMouseEvent * event )
```

Drag the object based on the mouse position in the overlay.

Parameters

<i>event</i>	The mouse event.
--------------	------------------

Returns

void

5.8.3.6 paintEvent()

```
void OverlayWidget::paintEvent (
    QPaintEvent * event ) [override], [protected]
```

Override the mousePressEvent method.

Parameters

<i>event</i>	The mouse event.
--------------	------------------

Returns

void

5.8.3.7 setActiveObject()

```
void OverlayWidget::setActiveObject (
    GameObject * obj ) [inline]
```

Get the active object.

Returns

GameObject* The active object.

5.8.3.8 setLastMousePos()

```
void OverlayWidget::setLastMousePos (
    QPoint pos ) [inline]
```

Get the last mouse position.

Returns

QPoint The last mouse position.

5.8.3.9 trySetSail()

```
void OverlayWidget::trySetSail (
    QMouseEvent * event )
```

Try grab the object based on the mouse position.

Parameters

<i>event</i>	The mouse event.
--------------	------------------

Returns

void

5.8.4 Member Data Documentation

5.8.4.1 activeObject

GameObject* OverlayWidget::activeObject [protected]

The active object.

5.8.4.2 graphView

QGraphicsView* OverlayWidget::graphView [protected]

The graphics view.

5.8.4.3 lastMousePos

```
QPoint OverlayWidget::lastMousePos [protected]
```

The last mouse position.

5.8.4.4 offset

```
QPoint OverlayWidget::offset [protected]
```

5.8.4.5 option

```
QStyleOptionGraphicsItem OverlayWidget::option [protected]
```

The option for the graphics item.

5.8.4.6 simEng

```
SimulationEngine* OverlayWidget::simEng [protected]
```

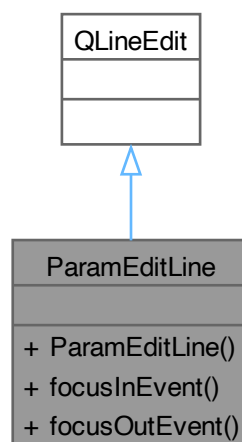
The simulation engine.

5.9 ParamEditLine Class Reference

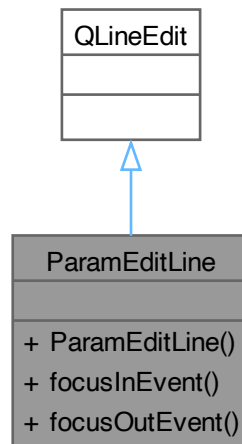
A class to represent a line edit widget for editing parameters.

```
#include <parameditline.hpp>
```

Inheritance diagram for ParamEditLine:



Collaboration diagram for ParamEditLine:



Signals

- void `focusIn` ()
Signal emitted when the line edit widget gains focus.
- void `focusOut` ()
Signal emitted when the line edit widget loses focus.

Public Member Functions

- `ParamEditLine` (`QWidget *parent=nullptr`)
Default constructor.
- void `focusInEvent` (`QFocusEvent *event`) override
Overridden focusInEvent method.
- void `focusOutEvent` (`QFocusEvent *event`) override
Overridden focusOutEvent method.

5.9.1 Detailed Description

A class to represent a line edit widget for editing parameters.

This class inherits from `QLineEdit` and provides a line edit widget for editing parameters.

See also

`QLineEdit`

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ParamEditLine()

```
ParamEditLine::ParamEditLine (
    QWidget * parent = nullptr ) [inline], [explicit]
```

Default constructor.

Parameters

<i>parent</i>	The parent widget.
---------------	--------------------

5.9.3 Member Function Documentation

5.9.3.1 focusIn

```
void ParamEditLine::focusIn ( ) [signal]
```

Signal emitted when the line edit widget gains focus.

Returns

void

5.9.3.2 focusInEvent()

```
void ParamEditLine::focusInEvent (
    QFocusEvent * event ) [inline], [override]
```

Overridden focusInEvent method.

Parameters

<i>event</i>	The focus event.
--------------	------------------

Returns

void

5.9.3.3 focusOut

```
void ParamEditLine::focusOut ( ) [signal]
```

Signal emitted when the line edit widget loses focus.

Returns

void

5.9.3.4 focusOutEvent()

```
void ParamEditLine::focusOutEvent (
    QFocusEvent * event ) [inline], [override]
```

Overridden focusOutEvent method.

Parameters

<i>event</i>	The focus event.
--------------	------------------

Returns

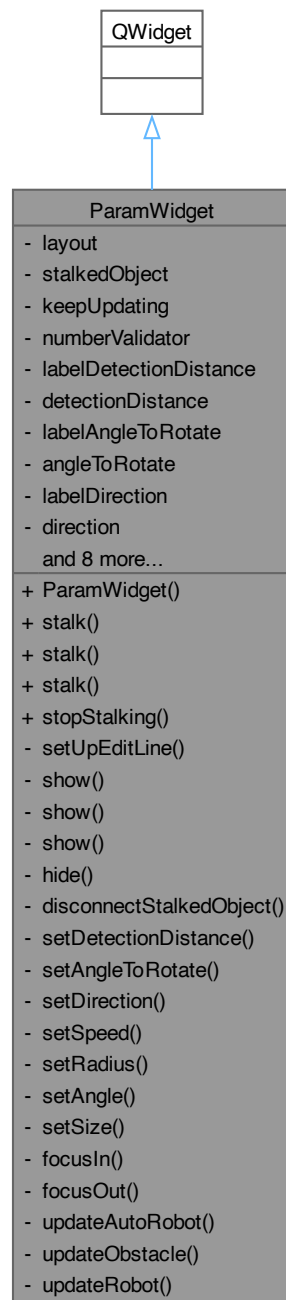
void

5.10 ParamWidget Class Reference

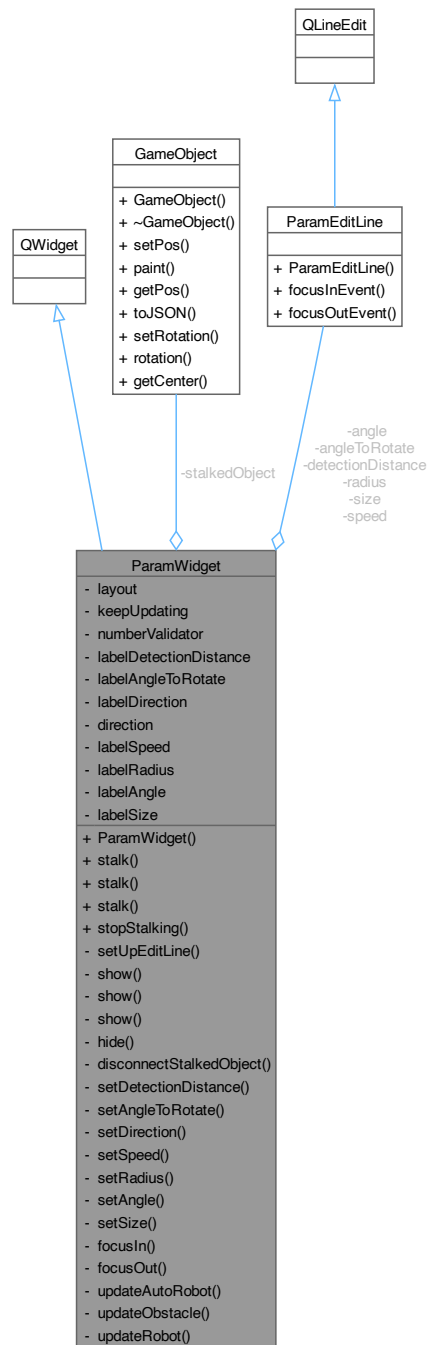
A class to represent a widget for editing parameters of game objects.

```
#include <paramwidget.hpp>
```

Inheritance diagram for ParamWidget:



Collaboration diagram for ParamWidget:



Public Member Functions

- **ParamWidget** (**QWidget** *parent=nullptr)
Default constructor.
- void **stalk** (**AutoRobot** *robot)
Set the game object whose parameters will be displayed.
- void **stalk** (**Obstacle** *obstacle)

- *Set the game object whose parameters will be displayed.*
void [stalk](#) ([Robot](#) *robot)
- *Set the game object whose parameters will be displayed.*
void [stopStalking](#) ()
- *Stop editing the parameters of the game object.*

Private Slots

- void [setDetectionDistance](#) ()
Signal to set the detection distance of the game object.
- void [setAngleToRotate](#) ()
Signal to set the angle to rotate of the game object.
- void [setDirection](#) ()
Signal to set the direction of the game object.
- void [setSpeed](#) ()
Signal to set the speed of the game object.
- void [setRadius](#) ()
Signal to set the radius of the game object.
- void [setAngle](#) ()
Signal to set the angle of the game object.
- void [setSize](#) ()
Signal to set the size of the game object.
- void [focusIn](#) ()
Signal to update the parameters of the game object.
- void [focusOut](#) ()
Signal to update the parameters of the game object.
- void [updateAutoRobot](#) ()
Update the parameters of the game object.
- void [updateObstacle](#) ()
Update the parameters of the game object.
- void [updateRobot](#) ()
Update the parameters of the game object.

Private Member Functions

- void [setUpEditLine](#) ([ParamEditLine](#) *lineEdit, QLabel *label)
Set up the line edit widget for editing a parameter.
- void [show](#) ([Robot](#) *robot)
Show the parameters of the game object.
- void [show](#) ([AutoRobot](#) *robot)
Show the parameters of the game object.
- void [show](#) ([Obstacle](#) *obstacle)
Show the parameters of the game object.
- void [hide](#) ()
Hide the widget.
- void [disconnectStalkedObject](#) ()
Disconnect the widget from the game object.

Private Attributes

- `QVBoxLayout * layout`
The layout of the widget.
- `GameObject * stalkedObject = nullptr`
The game object whose parameters are being displayed.
- `bool keepUpdating = true`
Whether the widget should keep updating the parameters of the game object.
- `QDoubleValidator * numberValidator`
The validator for the number input.
- `QLabel * labelDetectionDistance`
The labels and line edit widgets for editing the parameters.
- `ParamEditLine * detectionDistance`
- `QLabel * labelAngleToRotate`
- `ParamEditLine * angleToRotate`
- `QLabel * labelDirection`
- `QCheckBox * direction`
- `QLabel * labelSpeed`
- `ParamEditLine * speed`
- `QLabel * labelRadius`
- `ParamEditLine * radius`
- `QLabel * labelAngle`
- `ParamEditLine * angle`
- `QLabel * labelSize`
- `ParamEditLine * size`

5.10.1 Detailed Description

A class to represent a widget for editing parameters of game objects.

This class inherits from `QWidget` and provides a widget for editing parameters of game objects.

See also

`QWidget`

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ParamWidget()

```
ParamWidget::ParamWidget (
    QWidget * parent = nullptr ) [explicit]
```

Default constructor.

Parameters

<i>parent</i>	The parent widget.
---------------	--------------------

5.10.3 Member Function Documentation

5.10.3.1 disconnectStalkedObject()

```
void ParamWidget::disconnectStalkedObject ( ) [private]
```

Disconnect the widget from the game object.

Returns

void

5.10.3.2 focusIn

```
void ParamWidget::focusIn ( ) [inline], [private], [slot]
```

Signal to update the parameters of the game object.

Returns

void

5.10.3.3 focusOut

```
void ParamWidget::focusOut ( ) [inline], [private], [slot]
```

Signal to update the parameters of the game object.

Returns

void

5.10.3.4 hide()

```
void ParamWidget::hide ( ) [private]
```

Hide the widget.

Returns

void

5.10.3.5 setAngle

```
void ParamWidget::setAngle ( ) [private], [slot]
```

Signal to set the angle of the game object.

Returns

void

5.10.3.6 setAngleToRotate

```
void ParamWidget::setAngleToRotate ( ) [private], [slot]
```

Signal to set the angle to rotate of the game object.

Returns

void

5.10.3.7 setDetectionDistance

```
void ParamWidget::setDetectionDistance ( ) [private], [slot]
```

Signal to set the detection distance of the game object.

Returns

void

5.10.3.8 setDirection

```
void ParamWidget::setDirection ( ) [private], [slot]
```

Signal to set the direction of the game object.

Returns

void

5.10.3.9 setRadius

```
void ParamWidget::setRadius ( ) [private], [slot]
```

Signal to set the radius of the game object.

Returns

void

5.10.3.10 setSize

```
void ParamWidget::setSize ( ) [private], [slot]
```

Signal to set the size of the game object.

Returns

void

5.10.3.11 setSpeed

```
void ParamWidget::setSpeed ( ) [private], [slot]
```

Signal to set the speed of the game object.

Returns

void

5.10.3.12 setUpEditLine()

```
void ParamWidget::setUpEditLine (
    ParamEditLine * lineEdit,
    QLabel * label ) [private]
```

Set up the line edit widget for editing a parameter.

Parameters

<i>lineEdit</i>	The line edit widget.
<i>label</i>	The label for the line edit widget.

Returns

void

5.10.3.13 show() [1/3]

```
void ParamWidget::show (
    AutoRobot * robot ) [private]
```

Show the parameters of the game object.

Parameters

<i>robot</i>	The robot whose parameters will be displayed.
--------------	---

Returns

void

5.10.3.14 show() [2/3]

```
void ParamWidget::show (
    Obstacle * obstacle ) [private]
```

Show the parameters of the game object.

Parameters

<i>obstacle</i>	The obstacle whose parameters will be displayed.
-----------------	--

Returns

void

5.10.3.15 show() [3/3]

```
void ParamWidget::show (  
    Robot * robot ) [private]
```

Show the parameters of the game object.

Parameters

<i>robot</i>	The robot whose parameters will be displayed.
--------------	---

Returns

void

5.10.3.16 stalk() [1/3]

```
void ParamWidget::stalk (  
    AutoRobot * robot )
```

Set the game object whose parameters will be displayed.

Parameters

<i>object</i>	The game object.
---------------	------------------

Returns

void

5.10.3.17 stalk() [2/3]

```
void ParamWidget::stalk (  
    Obstacle * obstacle )
```

Set the game object whose parameters will be displayed.

Parameters

<i>object</i>	The game object.
---------------	------------------

Returns

void

5.10.3.18 stalk() [3/3]

```
void ParamWidget::stalk (
    Robot * robot )
```

Set the game object whose parameters will be displayed.

Parameters

<i>object</i>	The game object.
---------------	------------------

Returns

void

5.10.3.19 stopStalking()

```
void ParamWidget::stopStalking ( )
```

Stop editing the parameters of the game object.

Returns

void

5.10.3.20 updateAutoRobot

```
void ParamWidget::updateAutoRobot ( ) [private], [slot]
```

Update the parameters of the game object.

Returns

void

5.10.3.21 updateObstacle

```
void ParamWidget::updateObstacle ( ) [private], [slot]
```

Update the parameters of the game object.

Returns

void

5.10.3.22 updateRobot

```
void ParamWidget::updateRobot ( ) [private], [slot]
```

Update the parameters of the game object.

Returns

void

5.10.4 Member Data Documentation

5.10.4.1 angle

```
ParamEditLine* ParamWidget::angle [private]
```

5.10.4.2 angleToRotate

```
ParamEditLine* ParamWidget::angleToRotate [private]
```

5.10.4.3 detectionDistance

```
ParamEditLine* ParamWidget::detectionDistance [private]
```

5.10.4.4 direction

```
QCheckBox* ParamWidget::direction [private]
```

5.10.4.5 keepUpdating

```
bool ParamWidget::keepUpdating = true [private]
```

Whether the widget should keep updating the parameters of the game object.

5.10.4.6 labelAngle

```
QLabel* ParamWidget::labelAngle [private]
```

5.10.4.7 labelAngleToRotate

```
QLabel* ParamWidget::labelAngleToRotate [private]
```

5.10.4.8 labelDetectionDistance

```
QLabel* ParamWidget::labelDetectionDistance [private]
```

The labels and line edit widgets for editing the parameters.

5.10.4.9 labelDirection

```
QLabel* ParamWidget::labelDirection [private]
```

5.10.4.10 labelRadius

```
QLabel* ParamWidget::labelRadius [private]
```

5.10.4.11 labelSize

```
QLabel* ParamWidget::labelSize [private]
```

5.10.4.12 labelSpeed

```
QLabel* ParamWidget::labelSpeed [private]
```

5.10.4.13 layout

```
QVBoxLayout* ParamWidget::layout [private]
```

The layout of the widget.

5.10.4.14 numberValidator

```
QDoubleValidator* ParamWidget::numberValidator [private]
```

The validator for the number input.

5.10.4.15 radius

```
ParamEditLine* ParamWidget::radius [private]
```

5.10.4.16 size

```
ParamEditLine* ParamWidget::size [private]
```

5.10.4.17 speed

```
ParamEditLine* ParamWidget::speed [private]
```

5.10.4.18 stalkedObject

```
GameObject* ParamWidget::stalkedObject = nullptr [private]
```

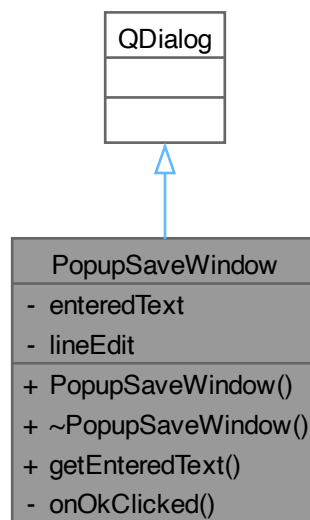
The game object whose parameters are being displayed.

5.11 PopupSaveWindow Class Reference

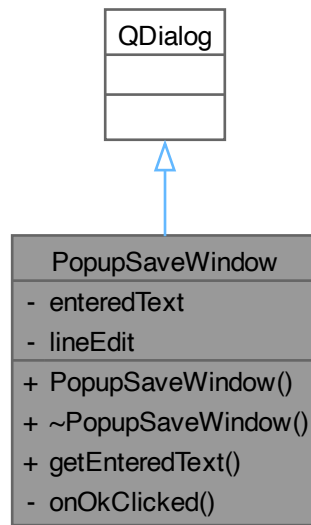
A class to represent a popup save window.

```
#include <popupsavewindow.h>
```

Inheritance diagram for PopupSaveWindow:



Collaboration diagram for PopupSaveWindow:



Public Member Functions

- [PopupSaveWindow](#) (QWidget *parent=nullptr)
Construct a new Popup Save Window object.
- [~PopupSaveWindow](#) ()
- QString [getEnteredText](#) ()
Get the entered text.

Private Slots

- void [onOkClicked](#) ()
Slot to handle the ok button click event.

Private Attributes

- QString [enteredText](#)
The entered text.
- QLineEdit * [lineEdit](#)
The line edit widget.

5.11.1 Detailed Description

A class to represent a popup save window.

This class provides an interface for creating and managing a popup save window.

See also

[QDialog](#)

5.11.2 Constructor & Destructor Documentation

5.11.2.1 PopupSaveWindow()

```
PopupSaveWindow::PopupSaveWindow (
    QWidget * parent = nullptr ) [explicit]
```

Construct a new Popup Save Window object.

Parameters

<i>parent</i>	The parent widget. Default is nullptr.
---------------	--

5.11.2.2 ~PopupSaveWindow()

```
PopupSaveWindow::~~PopupSaveWindow ( )
```

5.11.3 Member Function Documentation

5.11.3.1 getEnteredText()

```
QString PopupSaveWindow::getEnteredText ( ) [inline]
```

Get the entered text.

Returns

QString The entered text.

5.11.3.2 onOkClicked

```
void PopupSaveWindow::onOkClicked ( ) [private], [slot]
```

Slot to handle the ok button click event.

Returns

void

5.11.4 Member Data Documentation

5.11.4.1 enteredText

```
QString PopupSaveWindow::enteredText [private]
```

The entered text.

5.11.4.2 `lineEdit`

```
QLineEdit* PopupSaveWindow::lineEdit [private]
```

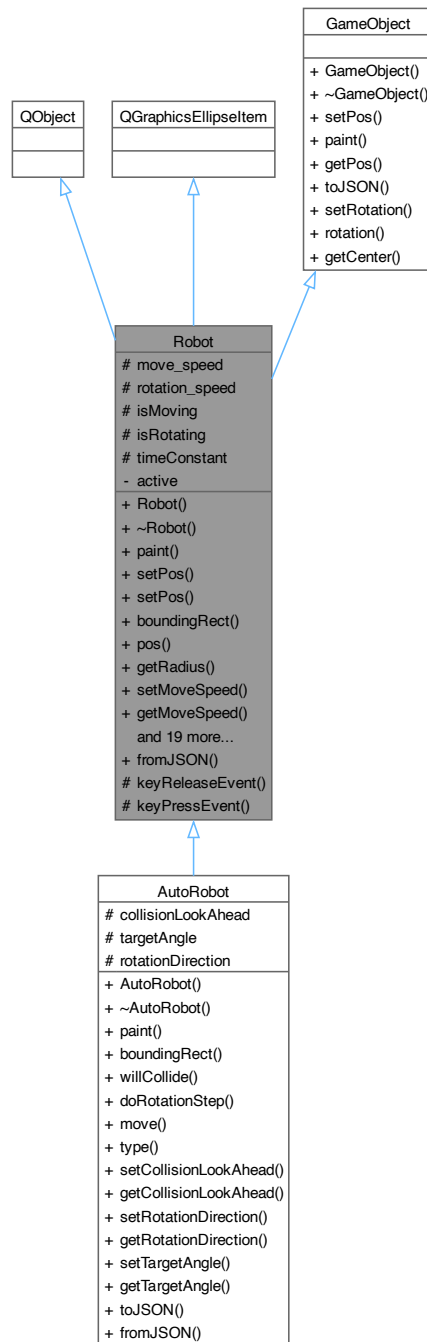
The line edit widget.

5.12 Robot Class Reference

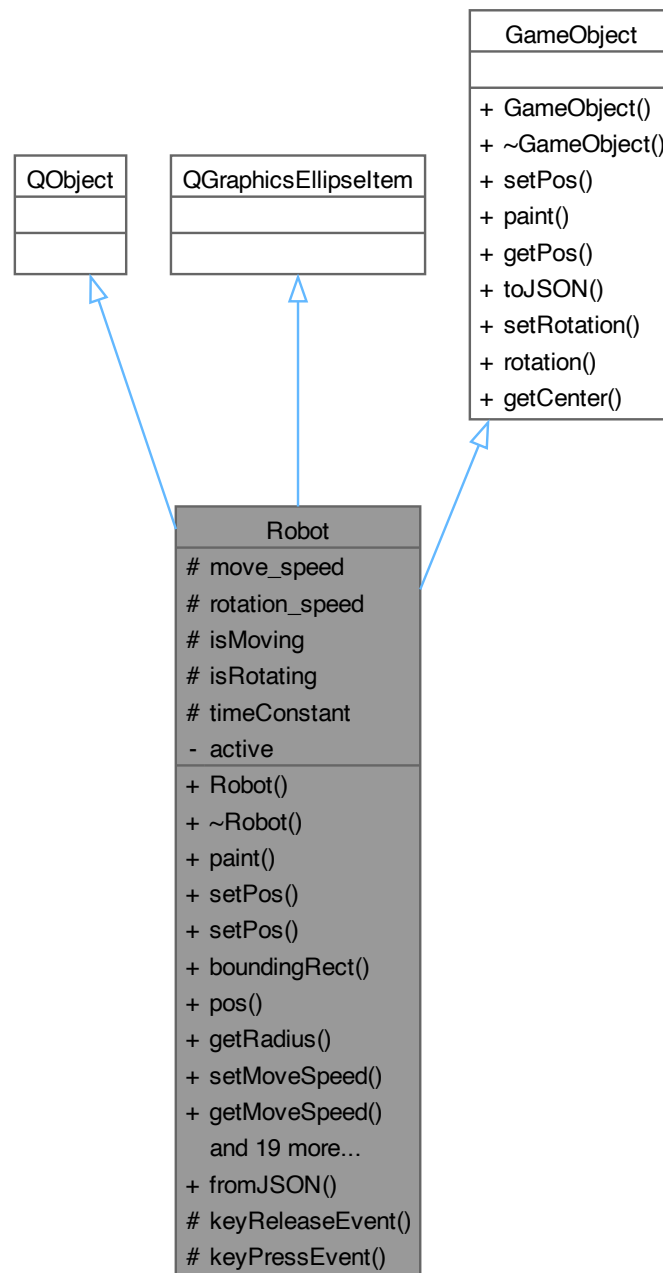
A class to represent a robot in the simulation. By default, the robot is a circle with a line drawn to represent its direction.

```
#include <robot.hpp>
```

Inheritance diagram for Robot:



Collaboration diagram for Robot:



Public Types

- enum `RotationDirection` { `Left` = -1 , `None` = 0 , `Right` = 1 }
- enum { `Type` = `QGraphicsItem::UserType` + 1 }

Enum to represent the direction of rotation of the robot.

Signals

- void `paramsUpdated` ()
Signal emitted when the parameters of the robot are updated.
- void `robotSepuku` ()
Signal emitted when the robot is removed.

Public Member Functions

- `Robot` (QGraphicsItem *parent=nullptr, qreal *timeConstant=nullptr)
Default constructor.
- `~Robot` ()
- virtual void `paint` (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override
- void `setPos` (const QPointF &pos)
- void `setPos` (qreal x, qreal y) override
- virtual QRectF `boundingRect` () const override
- QPointF `pos` ()
- qreal `getRadius` () const
- void `setMoveSpeed` (qreal speed)
Set the move speed of the robot.
- qreal `getMoveSpeed` ()
Get the move speed of the robot.
- void `setRotationSpeed` (qreal speed)
Set the rotation speed of the robot.
- qreal `getRotationSpeed` ()
Get the rotation speed of the robot.
- void `startMoving` ()
Allow the robot to be moved by setting the isMoving flag to true.
- void `stopMoving` ()
Stop the robot from moving by setting the isMoving flag to false.
- void `startRotating` (RotationDirection direction)
Start rotating the robot in the given direction.
- void `stopRotating` ()
Stop the robot from rotating by setting the isRotating flag to None.
- QPointF `getDirectionVector` ()
Get the direction vector of the robot.
- virtual bool `willCollide` (QPointF directionVector, qreal magnitude, bool allowAnticollision=false)
Check if the robot will collide with any other item in the scene or the scene boundaries if it moves by the given vector.
- virtual bool `move` ()
Move the robot based on its current direction and speed. Returns true if the robot moved, false if it didn't (e.g. if it hit a boundary).
- int `type` () const override
Get the type of the robot.
- QPointF `getPos` () override
Get the position of the robot.
- virtual QJsonObject `toJSON` () override
Convert the robot to a JSON object.
- void `toggleActive` ()
Toggle the active state of the robot.
- bool `isActive` ()
Check if the robot is active.

- qreal [getAngle](#) ()
Get the angle of the robot.
- void [setRadius](#) (qreal radius)
Set the angle of the robot.
- QPointF [getCenter](#) () override
Get the center of the robot.
- qreal [rotation](#) () override
Get the time constant of the simulation.
- void [setRotation](#) (qreal angle) override
Set the rotation of the robot.

Public Member Functions inherited from [GameObject](#)

- [GameObject](#) ()=default
- [~GameObject](#) ()=default

Static Public Member Functions

- static [Robot](#) * [fromJSON](#) (const QJsonObject &object, qreal *[timeConstant](#))
Create a [Robot](#) object from a JSON object.

Protected Member Functions

- void [keyReleaseEvent](#) (QKeyEvent *event)
The radius of the robot.
- void [keyPressEvent](#) (QKeyEvent *event)
Overridden keyPressEvent method.

Protected Attributes

- qreal [move_speed](#) = 1
The speed of the robot.
- qreal [rotation_speed](#) = 1
The speed of the rotation of the robot.
- bool [isMoving](#) = false
Flag to indicate if the robot is moving.
- [RotationDirection](#) [isRotating](#) = [RotationDirection::None](#)
Flag to indicate the direction of rotation.
- qreal * [timeConstant](#) = nullptr
The time constant of the simulation.

Private Attributes

- bool [active](#) = false
Flag to indicate if the robot is active.

5.12.1 Detailed Description

A class to represent a robot in the simulation. By default, the robot is a circle with a line drawn to represent its direction.

5.12.2 Member Enumeration Documentation

5.12.2.1 anonymous enum

anonymous enum

Enumerator

Type	
------	--

5.12.2.2 RotationDirection

enum `Robot::RotationDirection`

Enum to represent the direction of rotation of the robot.

Enumerator

Left	
None	
Right	

5.12.3 Constructor & Destructor Documentation

5.12.3.1 Robot()

```
Robot::Robot (
    QGraphicsItem * parent = nullptr,
    qreal * timeConstant = nullptr )
```

Default constructor.

Parameters

<i>parent</i>	The parent QGraphicsItem.
<i>timeConstant</i>	The time constant of the simulation.

Returns

void

The time constant is used to calculate the speed of the robot.

5.12.3.2 ~Robot()

```
Robot::~~Robot ( )
```

5.12.4 Member Function Documentation

5.12.4.1 boundingRect()

```
virtual QRectF Robot::boundingRect ( ) const [override], [virtual]
```

Reimplemented in [AutoRobot](#).

5.12.4.2 fromJSON()

```
static Robot * Robot::fromJSON (
    const QJsonObject & object,
    qreal * timeConstant ) [static]
```

Create a [Robot](#) object from a JSON object.

Parameters

<i>object</i>	The JSON object.
<i>timeConstant</i>	The time constant of the simulation.

Returns

Robot*

5.12.4.3 getAngle()

```
qreal Robot::getAngle ( ) [inline]
```

Get the angle of the robot.

Returns

qreal

5.12.4.4 getCenter()

```
QPointF Robot::getCenter ( ) [inline], [override], [virtual]
```

Get the center of the robot.

Returns

QPointF

Implements [GameObject](#).

5.12.4.5 `getDirectionVector()`

```
QPointF Robot::getDirectionVector ( )
```

Get the direction vector of the robot.

Returns

`QPointF` - Normalized vector representing the direction of the robot on the x and y axes

5.12.4.6 `getMoveSpeed()`

```
qreal Robot::getMoveSpeed ( )
```

Get the move speed of the robot.

Returns

`qreal`

5.12.4.7 `getPos()`

```
QPointF Robot::getPos ( ) [override], [virtual]
```

Get the position of the robot.

Returns

`QPointF`

Implements [GameObject](#).

5.12.4.8 `getRadius()`

```
qreal Robot::getRadius ( ) const
```

5.12.4.9 `getRotationSpeed()`

```
qreal Robot::getRotationSpeed ( )
```

Get the rotation speed of the robot.

Returns

`qreal`

5.12.4.10 isActive()

```
bool Robot::isActive ( ) [inline]
```

Check if the robot is active.

Returns

bool

5.12.4.11 keyPressEvent()

```
void Robot::keyPressEvent (
    QKeyEvent * event ) [protected]
```

Overridden keyPressEvent method.

This method is called when a key is pressed while the robot is focused.

Parameters

<i>event</i>	The key event.
--------------	----------------

Returns

void

5.12.4.12 keyReleaseEvent()

```
void Robot::keyReleaseEvent (
    QKeyEvent * event ) [protected]
```

The radius of the robot.

5.12.4.13 move()

```
virtual bool Robot::move ( ) [virtual]
```

Move the robot based on its current direction and speed. Returns true if the robot moved, false if it didn't (e.g. if it hit a boundary).

Returns

true

false

Reimplemented in [AutoRobot](#).

5.12.4.14 paint()

```
virtual void Robot::paint (
    QPainter * painter,
    const QStyleOptionGraphicsItem * option,
    QWidget * widget ) [override], [virtual]
```

Override the paint method to draw a line showing the direction of the robot

Implements [GameObject](#).

Reimplemented in [AutoRobot](#).

5.12.4.15 paramsUpdated

```
void Robot::paramsUpdated ( ) [signal]
```

Signal emitted when the parameters of the robot are updated.

Returns

void

5.12.4.16 pos()

```
QPointF Robot::pos ( )
```

Override pos to adjust to center-based positioning

5.12.4.17 robotSepuku

```
void Robot::robotSepuku ( ) [signal]
```

Signal emitted when the robot is removed.

Returns

void

5.12.4.18 rotation()

```
qreal Robot::rotation ( ) [inline], [override], [virtual]
```

Get the time constant of the simulation.

Returns

qreal

Implements [GameObject](#).

5.12.4.19 setMoveSpeed()

```
void Robot::setMoveSpeed (
    qreal speed )
```

Set the move speed of the robot.

Parameters

<i>speed</i>	
--------------	--

5.12.4.20 setPos() [1/2]

```
void Robot::setPos (
    const QPointF & pos )
```

Override setPos to adjust to center-based positioning

5.12.4.21 setPos() [2/2]

```
void Robot::setPos (
    qreal x,
    qreal y ) [override], [virtual]
```

Overload setPos to accept x and y coordinates

Implements [GameObject](#).

5.12.4.22 setRadius()

```
void Robot::setRadius (
    qreal radius )
```

Set the angle of the robot.

Parameters

<i>angle</i>	The angle to set.
--------------	-------------------

Returns

void

5.12.4.23 setRotation()

```
void Robot::setRotation (
    qreal angle ) [inline], [override], [virtual]
```

Set the rotation of the robot.

Parameters

<i>angle</i>	The angle to set.
--------------	-------------------

Returns

void

Implements [GameObject](#).

5.12.4.24 setRotationSpeed()

```
void Robot::setRotationSpeed (
    qreal speed )
```

Set the rotation speed of the robot.

Parameters

<i>speed</i>	
--------------	--

5.12.4.25 startMoving()

```
void Robot::startMoving ( )
```

Allow the robot to be moved by setting the isMoving flag to true.

5.12.4.26 startRotating()

```
void Robot::startRotating (
    RotationDirection direction )
```

Start rotating the robot in the given direction.

Parameters

<i>direction</i>	
------------------	--

5.12.4.27 stopMoving()

```
void Robot::stopMoving ( )
```

Stop the robot from moving by setting the isMoving flag to false.

5.12.4.28 stopRotating()

```
void Robot::stopRotating ( )
```

Stop the robot from rotating by setting the isRotating flag to None.

5.12.4.29 toggleActive()

```
void Robot::toggleActive ( ) [inline]
```

Toggle the active state of the robot.

If the robot is active, it will be drawn with a light gray fill. If it is inactive, it will be drawn with a transparent fill.

Returns

void

5.12.4.30 toJSON()

```
virtual QJsonObject Robot::toJSON ( ) [override], [virtual]
```

Convert the robot to a JSON object.

Returns

QJsonObject

Implements [GameObject](#).

Reimplemented in [AutoRobot](#).

5.12.4.31 type()

```
int Robot::type ( ) const [inline], [override]
```

Get the type of the robot.

Returns

int

5.12.4.32 willCollide()

```
virtual bool Robot::willCollide (
    QPointF directionVector,
    qreal magnitude,
    bool allowAnticollision = false ) [virtual]
```

Check if the robot will collide with any other item in the scene or the scene boundaries if it moves by the given vector.

Parameters

<i>moveVector</i>	The vector by which the robot will move
<i>allowAnticollision</i>	Flag to indicate if anticollision is allowed

Returns

`true` - if the robot will collide; `false` - if the robot will not collide

Reimplemented in [AutoRobot](#).

5.12.5 Member Data Documentation

5.12.5.1 active

```
bool Robot::active = false [private]
```

Flag to indicate if the robot is active.

5.12.5.2 isMoving

```
bool Robot::isMoving = false [protected]
```

Flag to indicate if the robot is moving.

5.12.5.3 isRotating

```
RotationDirection Robot::isRotating = RotationDirection::None [protected]
```

Flag to indicate the direction of rotation.

5.12.5.4 move_speed

```
qreal Robot::move_speed = 1 [protected]
```

The speed of the robot.

5.12.5.5 rotation_speed

```
qreal Robot::rotation_speed = 1 [protected]
```

The speed of the rotation of the robot.

5.12.5.6 timeConstant

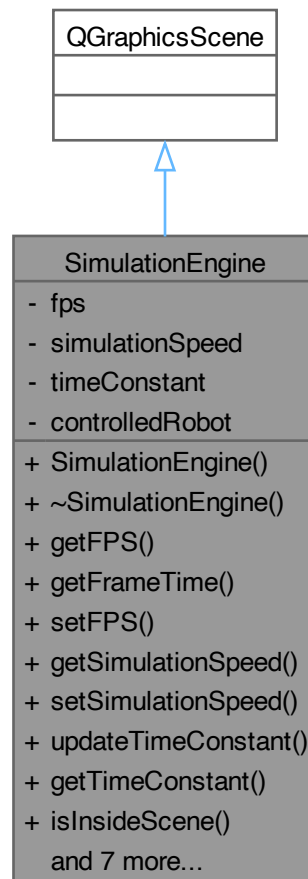
```
qreal* Robot::timeConstant = nullptr [protected]
```

The time constant of the simulation.

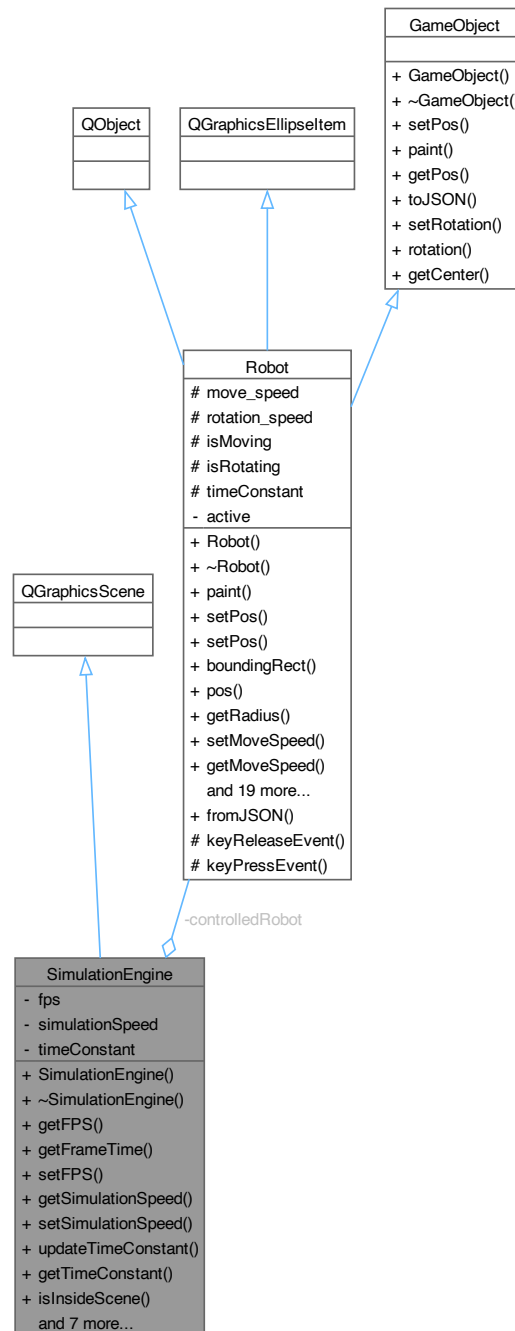
5.13 SimulationEngine Class Reference

```
#include <simulationengine.hpp>
```

Inheritance diagram for SimulationEngine:



Collaboration diagram for SimulationEngine:



Public Member Functions

- [SimulationEngine](#) (`QObject *parent=nullptr`, `int fps=60`, `qreal simulationSpeed=1.0/16.0`)
- [~SimulationEngine](#) ()
- `int` [getFPS](#) ()
Simulation Frames-Per-Second getter.
- `int` [getFrameTime](#) ()

- Get the time it takes to render a single frame.*

 - void `setFPS` (int `fps`)
- Set the simulation Frames-Per-Second.*

 - qreal `getSimulationSpeed` ()
- Get the simulation speed.*

 - void `setSimulationSpeed` (qreal `speed`)
- Set the simulation speed.*

 - void `updateTimeConstant` ()
- Update the time constant.*

 - qreal * `getTimeConstant` ()
- Get the time constant pointer.*

 - bool `isInsideScene` (const QPointF &`point`) const
- Check if a point is inside the scene.*

 - Robot * `getControlledRobot` ()
- Get the robot that is currently being controlled.*

 - void `setControlledRobot` (Robot *`robot`)
- Set the robot that is currently being controlled.*

 - bool `saveSimulation` (const QString &`filename`="simulation")
- Save the simulation.*

 - bool `loadSimulation` (QString `filename`="simulation")
- Load the simulation.*

 - void `read` (const QJsonObject &`json`)
- Read the simulation from a JSON object.*

 - QJsonObject `toJson` () const
- Convert the simulation to a JSON object.*

 - void `clearScene` ()
- Clear the scene.*

Private Attributes

- int `fps` = 60
- qreal `simulationSpeed` = 1
- qreal `timeConstant` = 1
- Robot * `controlledRobot` = nullptr

5.13.1 Constructor & Destructor Documentation

5.13.1.1 SimulationEngine()

```
SimulationEngine::SimulationEngine (
    QObject * parent = nullptr,
    int fps = 60,
    qreal simulationSpeed = 1.0/16.0 )
```

5.13.1.2 ~SimulationEngine()

```
SimulationEngine::~SimulationEngine ( )
```

5.13.2 Member Function Documentation

5.13.2.1 clearScene()

```
void SimulationEngine::clearScene ( )
```

Clear the scene.

5.13.2.2 getControlledRobot()

```
Robot * SimulationEngine::getControlledRobot ( )
```

Get the robot that is currently being controlled.

Returns

Robot*

5.13.2.3 getFPS()

```
int SimulationEngine::getFPS ( )
```

Simulation Frames-Per-Second getter.

Returns

int

5.13.2.4 getFrameTime()

```
int SimulationEngine::getFrameTime ( )
```

Get the time it takes to render a single frame.

Returns

int

5.13.2.5 getSimulationSpeed()

```
qreal SimulationEngine::getSimulationSpeed ( )
```

Get the simulation speed.

Returns

qreal

5.13.2.6 getTimeConstant()

```
qreal * SimulationEngine::getTimeConstant ( )
```

Get the time constant pointer.

Returns

qreal*

5.13.2.7 isInsideScene()

```
bool SimulationEngine::isInsideScene (
    const QPointF & point ) const
```

Check if a point is inside the scene.

Parameters

<i>point</i>	
--------------	--

Returns

bool

5.13.2.8 loadSimulation()

```
bool SimulationEngine::loadSimulation (
    QString filename = "simulation" )
```

Load the simulation.

Parameters

<i>filename</i>	The name of the file to load the simulation from.
-----------------	---

The file will be loaded from the JSON format from folders "simulations" and "exmaples"

Returns

void

5.13.2.9 read()

```
void SimulationEngine::read (
    const QJsonObject & json )
```

Read the simulation from a JSON object.

Parameters

<i>json</i>	The JSON object to read.
-------------	--------------------------

Returns

void

5.13.2.10 saveSimulation()

```
bool SimulationEngine::saveSimulation (
    const QString & filename = "simulation" )
```

Save the simulation.

Parameters

<i>filename</i>	The name of the file to save the simulation to.
-----------------	---

The file will be saved in the JSON format in folder "simulations"

Returns

void

5.13.2.11 setControlledRobot()

```
void SimulationEngine::setControlledRobot (
    Robot * robot )
```

Set the robot that is currently being controlled.

Parameters

<i>robot</i>	
--------------	--

Returns

void

5.13.2.12 setFPS()

```
void SimulationEngine::setFPS (
    int fps )
```

Set the simulation Frames-Per-Second.

Parameters

<i>fps</i>	
------------	--

5.13.2.13 setSimulationSpeed()

```
void SimulationEngine::setSimulationSpeed (
    qreal speed )
```

Set the simulation speed.

Parameters

<i>speed</i>	
--------------	--

Returns

void

5.13.2.14 toJson()

```
QJsonObject SimulationEngine::toJson ( ) const
```

Convert the simulation to a JSON object.

Returns

QJsonObject

5.13.2.15 updateTimeConstant()

```
void SimulationEngine::updateTimeConstant ( )
```

Update the time constant.

Returns

void

5.13.3 Member Data Documentation**5.13.3.1 controlledRobot**

```
Robot* SimulationEngine::controlledRobot = nullptr [private]
```

The robot that is currently being controlled.

5.13.3.2 fps

```
int SimulationEngine::fps = 60 [private]
```

The frames per second of the simulation engine.

5.13.3.3 simulationSpeed

```
qreal SimulationEngine::simulationSpeed = 1 [private]
```

The speed of the simulation engine.

5.13.3.4 timeConstant

```
qreal SimulationEngine::timeConstant = 1 [private]
```

The time constant of the simulation engine.

Index

- ~AutoRobot
 - AutoRobot, [15](#)
- ~GameObject
 - GameObject, [36](#)
- ~MainWindow
 - MainWindow, [42](#)
- ~Obstacle
 - Obstacle, [52](#)
- ~PopupSaveWindow
 - PopupSaveWindow, [79](#)
- ~Robot
 - Robot, [85](#)
- ~SimulationEngine
 - SimulationEngine, [96](#)
- active
 - Robot, [93](#)
- activeObject
 - OverlayWidget, [61](#)
- anchor
 - OverlayWidget, [59](#)
- angle
 - ParamWidget, [75](#)
- angleToRotate
 - ParamWidget, [75](#)
- AUTO
 - CheckableButton, [23](#)
- autoButton
 - ExpandableButtonWidget, [29](#)
- AutoRobot, [9](#)
 - ~AutoRobot, [15](#)
 - AutoRobot, [15](#)
 - boundingRect, [15](#)
 - collisionLookAhead, [19](#)
 - doRotationStep, [15](#)
 - fromJSON, [16](#)
 - getCollisionLookAhead, [16](#)
 - getRotationDirection, [16](#)
 - getTargetAngle, [16](#)
 - move, [17](#)
 - paint, [17](#)
 - rotationDirection, [19](#)
 - setCollisionLookAhead, [17](#)
 - setRotationDirection, [17](#)
 - setTargetAngle, [18](#)
 - targetAngle, [19](#)
 - toJSON, [18](#)
 - Type, [15](#)
 - type, [18](#)
 - willCollide, [18](#)
- boundingRect
 - AutoRobot, [15](#)
 - Robot, [86](#)
- CheckableButton, [20](#)
 - AUTO, [23](#)
 - CheckableButton, [23](#)
 - CONT, [23](#)
 - getOverlay, [23](#)
 - getWidgetPos, [23](#)
 - mouseMoveEvent, [24](#)
 - mousePressEvent, [24](#)
 - mouseReleaseEvent, [24](#)
 - ObjectType, [22](#)
 - objType, [25](#)
 - OBST, [23](#)
 - overlay, [25](#)
 - setOverlay, [25](#)
- clearScene
 - SimulationEngine, [97](#)
- collapse
 - ExpandableButtonWidget, [28](#)
- collisionLookAhead
 - AutoRobot, [19](#)
- CONT
 - CheckableButton, [23](#)
- controlButton
 - ExpandableButtonWidget, [29](#)
- controlledRobot
 - SimulationEngine, [100](#)
- convertFromRotatedSystem
 - OverlayWidget, [59](#)
- convertToRotatedSystem
 - OverlayWidget, [59](#)
- detectionDistance
 - ParamWidget, [75](#)
- direction
 - ParamWidget, [75](#)
- disconnectStalkedObject
 - ParamWidget, [70](#)
- doRotationStep
 - AutoRobot, [15](#)
- enteredText
 - PopupSaveWindow, [79](#)
- eventFilter
 - MainWindow, [42](#)
- expand
 - ExpandableButtonWidget, [29](#)

- ExpandableButtonWidget, 25
 - autoButton, 29
 - collapse, 28
 - controlButton, 29
 - expand, 29
 - ExpandableButtonWidget, 28
 - mainButton, 29
 - obstacleButton, 29
 - setOverlay, 29
- expandableWidget
 - MainWindow, 47
- ExpButton, 30
 - ExpButton, 31
 - mousePressEvent, 31
 - pressed, 33
- focusIn
 - ParamEditLine, 64
 - ParamWidget, 70
- focusInEvent
 - ParamEditLine, 64
- focusOut
 - ParamEditLine, 64
 - ParamWidget, 70
- focusOutEvent
 - ParamEditLine, 64
- fps
 - SimulationEngine, 100
- fromJSON
 - AutoRobot, 16
 - Obstacle, 52
 - Robot, 86
- GameObject, 33
 - ~GameObject, 36
 - GameObject, 36
 - getCenter, 36
 - getPos, 36
 - paint, 36
 - rotation, 37
 - setPos, 37
 - setRotation, 37
 - toJSON, 38
- getAngle
 - Robot, 86
- getCenter
 - GameObject, 36
 - Obstacle, 52
 - Robot, 86
- getCollisionLookAhead
 - AutoRobot, 16
- getControlledRobot
 - SimulationEngine, 97
- getDirectionVector
 - Robot, 86
- getEnteredText
 - PopupSaveWindow, 79
- getFPS
 - SimulationEngine, 97
- getFrameTime
 - SimulationEngine, 97
- getMoveSpeed
 - Robot, 87
- getOverlay
 - CheckableButton, 23
- getPos
 - GameObject, 36
 - Obstacle, 53
 - Robot, 87
- getRadius
 - Robot, 87
- getRotationDirection
 - AutoRobot, 16
- getRotationSpeed
 - Robot, 87
- getSimulationSpeed
 - SimulationEngine, 97
- getTargetAngle
 - AutoRobot, 16
- getTimeConstant
 - OverlayWidget, 60
 - SimulationEngine, 97
- getWidgetPos
 - CheckableButton, 23
- goLeft
 - MainWindow, 43
- goRight
 - MainWindow, 43
- goStraight
 - MainWindow, 43
- graphView
 - OverlayWidget, 61
- handleItemDoubleClick
 - MainWindow, 43
- hide
 - ParamWidget, 70
- initScene
 - MainWindow, 44
- isActive
 - Robot, 87
- isInsideScene
 - SimulationEngine, 98
- isMoving
 - Robot, 93
- isRotating
 - Robot, 93
- keepUpdating
 - ParamWidget, 75
- keyPressEvent
 - Robot, 88
- keyReleaseEvent
 - Robot, 88
- labelAngle
 - ParamWidget, 75

- labelAngleToRotate
 - ParamWidget, 76
- labelDetectionDistance
 - ParamWidget, 76
- labelDirection
 - ParamWidget, 76
- labelRadius
 - ParamWidget, 76
- labelSize
 - ParamWidget, 76
- labelSpeed
 - ParamWidget, 76
- lastMousePos
 - OverlayWidget, 61
- layout
 - ParamWidget, 76
- Left
 - Robot, 85
- lineEdit
 - PopupSaveWindow, 79
- listWidget
 - MainWindow, 47
- loadSimulation
 - SimulationEngine, 98
- mainButton
 - ExpandableButtonWidget, 29
- MainWindow, 38
 - ~MainWindow, 42
 - eventFilter, 42
 - expandableWidget, 47
 - goLeft, 43
 - goRight, 43
 - goStraight, 43
 - handleItemDoubleClick, 43
 - initScene, 44
 - listWidget, 47
 - MainWindow, 42
 - mouseDoubleClickEvent, 44
 - mouseMoveEvent, 44
 - mousePressEvent, 44
 - mouseReleaseEvent, 45
 - on_horizontalSlider_valueChanged, 45
 - on_pushButton_clicked, 45
 - overlay, 47
 - paramWidget, 48
 - resizeEvent, 45
 - saveSimulation, 46
 - setupAnimation, 46
 - showEvent, 46
 - simulationEngine, 48
 - stopMoving, 46
 - stopRotating, 47
 - toggleList, 47
 - ui, 48
 - updateAnimation, 47
- mouseDoubleClickEvent
 - MainWindow, 44
- mouseMoveEvent
 - CheckableButton, 24
 - MainWindow, 44
- mousePressEvent
 - CheckableButton, 24
 - ExpButton, 31
 - MainWindow, 44
- mouseReleaseEvent
 - CheckableButton, 24
 - MainWindow, 45
- move
 - AutoRobot, 17
 - Robot, 88
- move_speed
 - Robot, 93
- navigateTheSea
 - OverlayWidget, 60
- None
 - Robot, 85
- numberValidator
 - ParamWidget, 76
- ObjectType
 - CheckableButton, 22
- objType
 - CheckableButton, 25
- OBST
 - CheckableButton, 23
- Obstacle, 48
 - ~Obstacle, 52
 - fromJSON, 52
 - getCenter, 52
 - getPos, 53
 - Obstacle, 51, 52
 - obstacleSepuku, 53
 - paint, 53
 - paramsUpdated, 53
 - rotation, 54
 - setPos, 54
 - setRotation, 54
 - toJSON, 55
- obstacleButton
 - ExpandableButtonWidget, 29
- obstacleSepuku
 - Obstacle, 53
- offset
 - OverlayWidget, 62
- on_horizontalSlider_valueChanged
 - MainWindow, 45
- on_pushButton_clicked
 - MainWindow, 45
- onOkClicked
 - PopupSaveWindow, 79
- option
 - OverlayWidget, 62
- overlay
 - CheckableButton, 25
 - MainWindow, 47
- OverlayWidget, 55

- activeObject, 61
- anchor, 59
- convertFromRotatedSystem, 59
- convertToRotatedSystem, 59
- getTimeConstant, 60
- graphView, 61
- lastMousePos, 61
- navigateTheSea, 60
- offset, 62
- option, 62
- OverlayWidget, 58
- paintEvent, 60
- setActiveObject, 60
- setLastMousePos, 61
- simEng, 62
- trySetSail, 61
- paint
 - AutoRobot, 17
 - GameObject, 36
 - Obstacle, 53
 - Robot, 88
- paintEvent
 - OverlayWidget, 60
- ParamEditLine, 62
 - focusIn, 64
 - focusInEvent, 64
 - focusOut, 64
 - focusOutEvent, 64
 - ParamEditLine, 63
- paramsUpdated
 - Obstacle, 53
 - Robot, 89
- ParamWidget, 65
 - angle, 75
 - angleToRotate, 75
 - detectionDistance, 75
 - direction, 75
 - disconnectStalkedObject, 70
 - focusIn, 70
 - focusOut, 70
 - hide, 70
 - keepUpdating, 75
 - labelAngle, 75
 - labelAngleToRotate, 76
 - labelDetectionDistance, 76
 - labelDirection, 76
 - labelRadius, 76
 - labelSize, 76
 - labelSpeed, 76
 - layout, 76
 - numberValidator, 76
 - ParamWidget, 69
 - radius, 76
 - setAngle, 70
 - setAngleToRotate, 70
 - setDetectionDistance, 71
 - setDirection, 71
 - setRadius, 71
 - setSize, 71
 - setSpeed, 71
 - setUpEditLine, 72
 - show, 72, 73
 - size, 77
 - speed, 77
 - stalk, 73, 74
 - stalkedObject, 77
 - stopStalking, 74
 - updateAutoRobot, 74
 - updateObstacle, 74
 - updateRobot, 75
- paramWidget
 - MainWindow, 48
- PopupSaveWindow, 77
 - ~PopupSaveWindow, 79
 - enteredText, 79
 - getEnteredText, 79
 - lineEdit, 79
 - onOkClicked, 79
 - PopupSaveWindow, 79
- pos
 - Robot, 89
- pressed
 - ExpButton, 33
- radius
 - ParamWidget, 76
- read
 - SimulationEngine, 98
- resizeEvent
 - MainWindow, 45
- Right
 - Robot, 85
- Robot, 80
 - ~Robot, 85
 - active, 93
 - boundingRect, 86
 - fromJSON, 86
 - getAngle, 86
 - getCenter, 86
 - getDirectionVector, 86
 - getMoveSpeed, 87
 - getPos, 87
 - getRadius, 87
 - getRotationSpeed, 87
 - isActive, 87
 - isMoving, 93
 - isRotating, 93
 - keyPressEvent, 88
 - keyReleaseEvent, 88
 - Left, 85
 - move, 88
 - move_speed, 93
 - None, 85
 - paint, 88
 - paramsUpdated, 89
 - pos, 89
 - Right, 85

- Robot, 85
- robotSepuku, 89
- rotation, 89
- rotation_speed, 93
- RotationDirection, 85
- setMoveSpeed, 89
- setPos, 90
- setRadius, 90
- setRotation, 90
- setRotationSpeed, 91
- startMoving, 91
- startRotating, 91
- stopMoving, 91
- stopRotating, 91
- timeConstant, 93
- toggleActive, 91
- toJSON, 92
- Type, 85
- type, 92
- willCollide, 92
- robotSepuku
 - Robot, 89
- rotation
 - GameObject, 37
 - Obstacle, 54
 - Robot, 89
- rotation_speed
 - Robot, 93
- RotationDirection
 - Robot, 85
- rotationDirection
 - AutoRobot, 19
- saveSimulation
 - MainWindow, 46
 - SimulationEngine, 99
- setActiveObject
 - OverlayWidget, 60
- setAngle
 - ParamWidget, 70
- setAngleToRotate
 - ParamWidget, 70
- setCollisionLookAhead
 - AutoRobot, 17
- setControlledRobot
 - SimulationEngine, 99
- setDetectionDistance
 - ParamWidget, 71
- setDirection
 - ParamWidget, 71
- setFPS
 - SimulationEngine, 99
- setLastMousePos
 - OverlayWidget, 61
- setMoveSpeed
 - Robot, 89
- setOverlay
 - CheckableButton, 25
 - ExpandableButtonWidget, 29
- setPos
 - GameObject, 37
 - Obstacle, 54
 - Robot, 90
- setRadius
 - ParamWidget, 71
 - Robot, 90
- setRotation
 - GameObject, 37
 - Obstacle, 54
 - Robot, 90
- setRotationDirection
 - AutoRobot, 17
- setRotationSpeed
 - Robot, 91
- setSimulationSpeed
 - SimulationEngine, 100
- setSize
 - ParamWidget, 71
- setSpeed
 - ParamWidget, 71
- setTargetAngle
 - AutoRobot, 18
- setUpAnimation
 - MainWindow, 46
- setUpEditLine
 - ParamWidget, 72
- show
 - ParamWidget, 72, 73
- showEvent
 - MainWindow, 46
- simEng
 - OverlayWidget, 62
- SimulationEngine, 94
 - ~SimulationEngine, 96
 - clearScene, 97
 - controlledRobot, 100
 - fps, 100
 - getControlledRobot, 97
 - getFPS, 97
 - getFrameTime, 97
 - getSimulationSpeed, 97
 - getTimeConstant, 97
 - isInsideScene, 98
 - loadSimulation, 98
 - read, 98
 - saveSimulation, 99
 - setControlledRobot, 99
 - setFPS, 99
 - setSimulationSpeed, 100
 - SimulationEngine, 96
 - simulationSpeed, 101
 - timeConstant, 101
 - toJson, 100
 - updateTimeConstant, 100
- simulationEngine
 - MainWindow, 48
- simulationSpeed

- SimulationEngine, [101](#)
- size
 - ParamWidget, [77](#)
- speed
 - ParamWidget, [77](#)
- stalk
 - ParamWidget, [73](#), [74](#)
- stalkedObject
 - ParamWidget, [77](#)
- startMoving
 - Robot, [91](#)
- startRotating
 - Robot, [91](#)
- stopMoving
 - MainWindow, [46](#)
 - Robot, [91](#)
- stopRotating
 - MainWindow, [47](#)
 - Robot, [91](#)
- stopStalking
 - ParamWidget, [74](#)
- targetAngle
 - AutoRobot, [19](#)
- timeConstant
 - Robot, [93](#)
 - SimulationEngine, [101](#)
- toggleActive
 - Robot, [91](#)
- toggleList
 - MainWindow, [47](#)
- toJSON
 - AutoRobot, [18](#)
 - GameObject, [38](#)
 - Obstacle, [55](#)
 - Robot, [92](#)
- toJson
 - SimulationEngine, [100](#)
- trySetSail
 - OverlayWidget, [61](#)
- Type
 - AutoRobot, [15](#)
 - Robot, [85](#)
- type
 - AutoRobot, [18](#)
 - Robot, [92](#)
- Ui, [7](#)
- ui
 - MainWindow, [48](#)
- updateAnimation
 - MainWindow, [47](#)
- updateAutoRobot
 - ParamWidget, [74](#)
- updateObstacle
 - ParamWidget, [74](#)
- updateRobot
 - ParamWidget, [75](#)
- updateTimeConstant

- SimulationEngine, [100](#)
- willCollide
 - AutoRobot, [18](#)
 - Robot, [92](#)