

ICP Project 2023/2024

1.0

Generated by Doxygen 1.9.8

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 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 Ui Namespace Reference	9
6 Class Documentation	11
6.1 AutoRobot Class Reference	11
6.1.1 Detailed Description	16
6.1.2 Member Enumeration Documentation	16
6.1.2.1 anonymous enum	16
6.1.3 Constructor & Destructor Documentation	17
6.1.3.1 AutoRobot()	17
6.1.3.2 ~AutoRobot()	17
6.1.4 Member Function Documentation	17
6.1.4.1 boundingRect()	17
6.1.4.2 doRotationStep()	17
6.1.4.3 fromJSON()	18
6.1.4.4 getCollisionLookAhead()	18
6.1.4.5 getRotationDirection()	18
6.1.4.6 getTargetAngle()	19
6.1.4.7 move()	19
6.1.4.8 paint()	19
6.1.4.9 setCollisionLookAhead()	19
6.1.4.10 setRotationDirection()	20
6.1.4.11 setTargetAngle()	20
6.1.4.12 toJSON()	20
6.1.4.13 type()	21
6.1.4.14 willCollide()	21
6.1.5 Member Data Documentation	21
6.1.5.1 collisionLookAhead	21
6.1.5.2 rotationDirection	21
6.1.5.3 targetAngle	22
6.2 CheckableButton Class Reference	22
6.2.1 Detailed Description	24

6.2.2 Member Enumeration Documentation	24
6.2.2.1 ObjectType	24
6.2.3 Constructor & Destructor Documentation	25
6.2.3.1 CheckableButton()	25
6.2.4 Member Function Documentation	25
6.2.4.1 getOverlay()	25
6.2.4.2 getWidgetPos()	25
6.2.4.3 mouseMoveEvent	26
6.2.4.4 mousePressEvent	26
6.2.4.5 mouseReleaseEvent	26
6.2.4.6 setOverlay()	27
6.2.5 Member Data Documentation	27
6.2.5.1 objType	27
6.2.5.2 overlay	27
6.3 ExpandableButtonWidget Class Reference	28
6.3.1 Detailed Description	30
6.3.2 Constructor & Destructor Documentation	30
6.3.2.1 ExpandableButtonWidget()	30
6.3.3 Member Function Documentation	30
6.3.3.1 collapse()	30
6.3.3.2 expand	31
6.3.3.3 setOverlay()	31
6.3.4 Member Data Documentation	31
6.3.4.1 autoButton	31
6.3.4.2 controlButton	31
6.3.4.3 mainButton	31
6.3.4.4 obstacleButton	32
6.4 ExpButton Class Reference	32
6.4.1 Detailed Description	33
6.4.2 Constructor & Destructor Documentation	34
6.4.2.1 ExpButton()	34
6.4.3 Member Function Documentation	35
6.4.3.1 mousePressEvent	35
6.4.3.2 pressed	35
6.5 GameObject Class Reference	35
6.5.1 Detailed Description	37
6.5.2 Constructor & Destructor Documentation	38
6.5.2.1 GameObject()	38
6.5.2.2 ~GameObject()	38
6.5.3 Member Function Documentation	38
6.5.3.1 getCenter()	38
6.5.3.2 getPos()	38

6.5.3.3 paint()	38
6.5.3.4 rotation()	39
6.5.3.5 setPos()	39
6.5.3.6 setRotation()	39
6.5.3.7 toJSON()	40
6.6 MainWindow Class Reference	40
6.6.1 Detailed Description	44
6.6.2 Constructor & Destructor Documentation	44
6.6.2.1 MainWindow()	44
6.6.2.2 ~MainWindow()	44
6.6.3 Member Function Documentation	44
6.6.3.1 eventFilter	44
6.6.3.2 goLeft	45
6.6.3.3 goRight	45
6.6.3.4 goStraight	45
6.6.3.5 handleItemDoubleClick	45
6.6.3.6 initScene()	46
6.6.3.7 mouseDoubleClickEvent	46
6.6.3.8 mouseMoveEvent	46
6.6.3.9 mousePressEvent	46
6.6.3.10 mouseReleaseEvent	47
6.6.3.11 on_horizontalSlider_valueChanged	47
6.6.3.12 on_pushButton_clicked	47
6.6.3.13 resizeEvent	48
6.6.3.14 saveSimulation	48
6.6.3.15 setupAnimation()	48
6.6.3.16 showEvent	48
6.6.3.17 stopMoving	48
6.6.3.18 stopRotating	49
6.6.3.19 toggleList	49
6.6.3.20 updateAnimation()	49
6.6.4 Member Data Documentation	49
6.6.4.1 expandableWidget	49
6.6.4.2 listWidget	49
6.6.4.3 overlay	50
6.6.4.4 paramWidget	50
6.6.4.5 simulationEngine	50
6.6.4.6 ui	50
6.7 Obstacle Class Reference	51
6.7.1 Detailed Description	53
6.7.2 Constructor & Destructor Documentation	53
6.7.2.1 Obstacle() [1/2]	53

6.7.2.2 Obstacle() [2/2]	54
6.7.2.3 ~Obstacle()	54
6.7.3 Member Function Documentation	54
6.7.3.1 fromJson()	54
6.7.3.2 getCenter()	55
6.7.3.3 getPos()	55
6.7.3.4 obstacleSepuku	55
6.7.3.5 paint()	55
6.7.3.6 paramsUpdated	56
6.7.3.7 rotation()	56
6.7.3.8 setPos()	56
6.7.3.9 setRotation()	57
6.7.3.10 toJson()	57
6.8 OverlayWidget Class Reference	57
6.8.1 Detailed Description	60
6.8.2 Constructor & Destructor Documentation	60
6.8.2.1 OverlayWidget()	60
6.8.3 Member Function Documentation	61
6.8.3.1 anchor()	61
6.8.3.2 convertFromRotatedSystem()	61
6.8.3.3 convertToRotatedSystem()	61
6.8.3.4 getTimeConstant()	62
6.8.3.5 navigateTheSea()	62
6.8.3.6 paintEvent()	62
6.8.3.7 setActiveObject()	63
6.8.3.8 setLastMousePos()	63
6.8.3.9 trySetSail()	63
6.8.4 Member Data Documentation	63
6.8.4.1 activeObject	63
6.8.4.2 graphView	64
6.8.4.3 lastMousePos	64
6.8.4.4 offset	64
6.8.4.5 option	64
6.8.4.6 simEng	64
6.9 ParamEditLine Class Reference	65
6.9.1 Detailed Description	66
6.9.2 Constructor & Destructor Documentation	66
6.9.2.1 ParamEditLine()	66
6.9.3 Member Function Documentation	67
6.9.3.1 focusIn	67
6.9.3.2 focusInEvent()	67
6.9.3.3 focusOut	67

6.9.3.4 focusOutEvent()	67
6.10 ParamWidget Class Reference	68
6.10.1 Detailed Description	72
6.10.2 Constructor & Destructor Documentation	72
6.10.2.1 ParamWidget()	72
6.10.3 Member Function Documentation	72
6.10.3.1 disconnectStalkedObject()	72
6.10.3.2 focusIn	73
6.10.3.3 focusOut	73
6.10.3.4 hide()	73
6.10.3.5 setAngle	73
6.10.3.6 setAngleToRotate	74
6.10.3.7 setDetectionDistance	74
6.10.3.8 setDirection	74
6.10.3.9 setRadius	74
6.10.3.10 setSize	74
6.10.3.11 setSpeed	75
6.10.3.12 setUpEditLine()	75
6.10.3.13 show() [1/3]	75
6.10.3.14 show() [2/3]	75
6.10.3.15 show() [3/3]	76
6.10.3.16 stalk() [1/3]	76
6.10.3.17 stalk() [2/3]	76
6.10.3.18 stalk() [3/3]	77
6.10.3.19 stopStalking()	77
6.10.3.20 updateAutoRobot	77
6.10.3.21 updateObstacle	78
6.10.3.22 updateRobot	78
6.10.4 Member Data Documentation	78
6.10.4.1 angle	78
6.10.4.2 angleToRotate	78
6.10.4.3 detectionDistance	78
6.10.4.4 direction	78
6.10.4.5 keepUpdating	79
6.10.4.6 labelAngle	79
6.10.4.7 labelAngleToRotate	79
6.10.4.8 labelDetectionDistance	79
6.10.4.9 labelDirection	79
6.10.4.10 labelRadius	79
6.10.4.11 labelSize	79
6.10.4.12 labelSpeed	80
6.10.4.13 layout	80

6.10.4.14 numberValidator	80
6.10.4.15 radius	80
6.10.4.16 size	80
6.10.4.17 speed	80
6.10.4.18 stalkedObject	80
6.11 PopupSaveWindow Class Reference	81
6.11.1 Detailed Description	82
6.11.2 Constructor & Destructor Documentation	82
6.11.2.1 PopupSaveWindow()	82
6.11.2.2 ~PopupSaveWindow()	83
6.11.3 Member Function Documentation	83
6.11.3.1 getEnteredText()	83
6.11.3.2 onOkClicked	83
6.11.4 Member Data Documentation	83
6.11.4.1 enteredText	83
6.11.4.2 lineEdit	83
6.12 QDialog Class Reference	84
6.13 QGraphicsEllipseItem Class Reference	85
6.14 QGraphicsRectItem Class Reference	86
6.15 QGraphicsScene Class Reference	88
6.16 QLineEdit Class Reference	89
6.17 QMainWindow Class Reference	90
6.18 QObject Class Reference	92
6.19 QPushButton Class Reference	93
6.20 QWidget Class Reference	95
6.21 Robot Class Reference	96
6.21.1 Detailed Description	100
6.21.2 Member Enumeration Documentation	100
6.21.2.1 anonymous enum	100
6.21.2.2 RotationDirection	101
6.21.3 Constructor & Destructor Documentation	101
6.21.3.1 Robot()	101
6.21.3.2 ~Robot()	101
6.21.4 Member Function Documentation	102
6.21.4.1 boundingRect()	102
6.21.4.2 fromJSON()	102
6.21.4.3 getAngle()	102
6.21.4.4 getCenter()	102
6.21.4.5 getDirectionVector()	103
6.21.4.6 getMoveSpeed()	103
6.21.4.7 getPos()	103
6.21.4.8 getRadius()	103

6.21.4.9 getRotationSpeed()	103
6.21.4.10 isActive()	104
6.21.4.11 keyPressEvent()	104
6.21.4.12 keyReleaseEvent()	104
6.21.4.13 move()	104
6.21.4.14 paint()	105
6.21.4.15 paramsUpdated	105
6.21.4.16 pos()	105
6.21.4.17 robotSepuku	105
6.21.4.18 rotation()	105
6.21.4.19 setMoveSpeed()	105
6.21.4.20 setPos() [1/2]	106
6.21.4.21 setPos() [2/2]	106
6.21.4.22 setRadius()	106
6.21.4.23 setRotation()	106
6.21.4.24 setRotationSpeed()	107
6.21.4.25 startMoving()	107
6.21.4.26 startRotating()	107
6.21.4.27 stopMoving()	107
6.21.4.28 stopRotating()	107
6.21.4.29 toggleActive()	108
6.21.4.30 toJSON()	108
6.21.4.31 type()	108
6.21.4.32 willCollide()	108
6.21.5 Member Data Documentation	109
6.21.5.1 active	109
6.21.5.2 isMoving	109
6.21.5.3 isRotating	109
6.21.5.4 move_speed	109
6.21.5.5 rotation_speed	110
6.21.5.6 timeConstant	110
6.22 SimulationEngine Class Reference	110
6.22.1 Detailed Description	112
6.22.2 Constructor & Destructor Documentation	112
6.22.2.1 SimulationEngine()	112
6.22.2.2 ~SimulationEngine()	112
6.22.3 Member Function Documentation	112
6.22.3.1 clearScene()	112
6.22.3.2 getControlledRobot()	113
6.22.3.3 getFPS()	113
6.22.3.4 getFrameTime()	113
6.22.3.5 getSimulationSpeed()	113

6.22.3.6	getTimeConstant()	113
6.22.3.7	isInsideScene()	113
6.22.3.8	loadSimulation()	114
6.22.3.9	read()	114
6.22.3.10	saveSimulation()	114
6.22.3.11	setControlledRobot()	115
6.22.3.12	setFPS()	115
6.22.3.13	setSimulationSpeed()	115
6.22.3.14	toJson()	116
6.22.3.15	updateTimeConstant()	116
6.22.4	Member Data Documentation	116
6.22.4.1	controlledRobot	116
6.22.4.2	fps	116
6.22.4.3	simulationSpeed	116
6.22.4.4	timeConstant	116
7	File Documentation	117
7.1	autorobot.hpp File Reference	117
7.1.1	Detailed Description	118
7.1.2	Macro Definition Documentation	119
7.1.2.1	SMOOTH_ROTATION_SPEED	119
7.2	autorobot.hpp	119
7.3	checkablebutton.hpp File Reference	120
7.3.1	Detailed Description	121
7.4	checkablebutton.hpp	122
7.5	expbutton.hpp File Reference	123
7.5.1	Detailed Description	124
7.6	expbutton.hpp	125
7.7	expbuttonwidget.hpp File Reference	125
7.7.1	Detailed Description	126
7.8	expbuttonwidget.hpp	127
7.9	gameobject.hpp File Reference	127
7.9.1	Detailed Description	128
7.10	gameobject.hpp	129
7.11	mainwindow.h File Reference	130
7.11.1	Detailed Description	131
7.12	mainwindow.h	131
7.13	obstacle.hpp File Reference	133
7.13.1	Detailed Description	134
7.14	obstacle.hpp	135
7.15	overlaywidget.hpp File Reference	136
7.15.1	Detailed Description	137

7.16 overlaywidget.hpp	138
7.17 parameditline.hpp File Reference	139
7.17.1 Detailed Description	140
7.18 parameditline.hpp	141
7.19 paramwidget.hpp File Reference	141
7.19.1 Detailed Description	142
7.20 paramwidget.hpp	143
7.21 popupsavewindow.h File Reference	145
7.21.1 Detailed Description	146
7.22 popupsavewindow.h	146
7.23 robot.hpp File Reference	147
7.23.1 Detailed Description	148
7.23.2 Macro Definition Documentation	149
7.23.2.1 BODY_COLLISION_MARGIN	149
7.24 robot.hpp	149
7.25 simulationengine.hpp File Reference	152
7.25.1 Detailed Description	153
7.26 simulationengine.hpp	154
Index	157

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Ui	9
------------------------------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

GameObject	35
Obstacle	51
Robot	96
AutoRobot	11
QDialog	84
PopupSaveWindow	81
QGraphicsEllipseItem	85
Robot	96
QGraphicsRectItem	86
Obstacle	51
QGraphicsScene	88
SimulationEngine	110
QLineEdit	89
ParamEditLine	65
QMainWindow	90
MainWindow	40
QObject	92
Obstacle	51
Robot	96
QPushButton	93
CheckableButton	22
ExpButton	32
QWidget	95
ExpandableButtonWidget	28
OverlayWidget	57
ParamWidget	68

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	11
CheckableButton	
A class to represent a checkable button	22
ExpandableButtonWidget	
A class to represent an expandable button widget	28
ExpButton	
A class for expandable buttons	32
GameObject	
A class to represent a game object in the simulation	35
MainWindow	
A class to represent the main window of the application	40
Obstacle	
A class to represent an obstacle	51
OverlayWidget	
A class to represent an overlay widget	57
ParamEditLine	
A class to represent a line edit widget for editing parameters	65
ParamWidget	
A class to represent a widget for editing parameters of game objects	68
PopupSaveWindow	
A class to represent a popup save window	81
QDialog	84
QGraphicsEllipseItem	85
QGraphicsRectItem	86
QGraphicsScene	88
QLineEdit	89
QMainWindow	90
QObject	92
QPushButton	93
QWidget	95
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	96
SimulationEngine	110

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

autorobot.hpp	This file contains the declaration of the AutoRobot class	117
checkablebutton.hpp	This file contains the declaration of the CheckableButton class	120
expbutton.hpp	This file contains the declaration of the ExpButton class	123
expbuttonwidget.hpp	This file contains the declaration of the ExpandableButtonWidget class	125
gameobject.hpp	This file contains the declaration of the GameObject class	127
mainwindow.h	This file contains the declaration of the MainWindow class	130
obstacle.hpp	This file contains the declaration of the Obstacle class	133
overlaywidget.hpp	This file contains the declaration of the OverlayWidget class	136
parameditline.hpp	This file contains the declaration of the ParamEditLine class	139
paramwidget.hpp	This file contains the declaration of the ParamWidget class	141
popsavesavewindow.h	This file contains the declaration of the PopupSaveWindow class	145
robot.hpp	This file contains the declaration of the Robot class	147
simulationengine.hpp	This file contains the declaration of the SimulationEngine class	152

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

Chapter 6

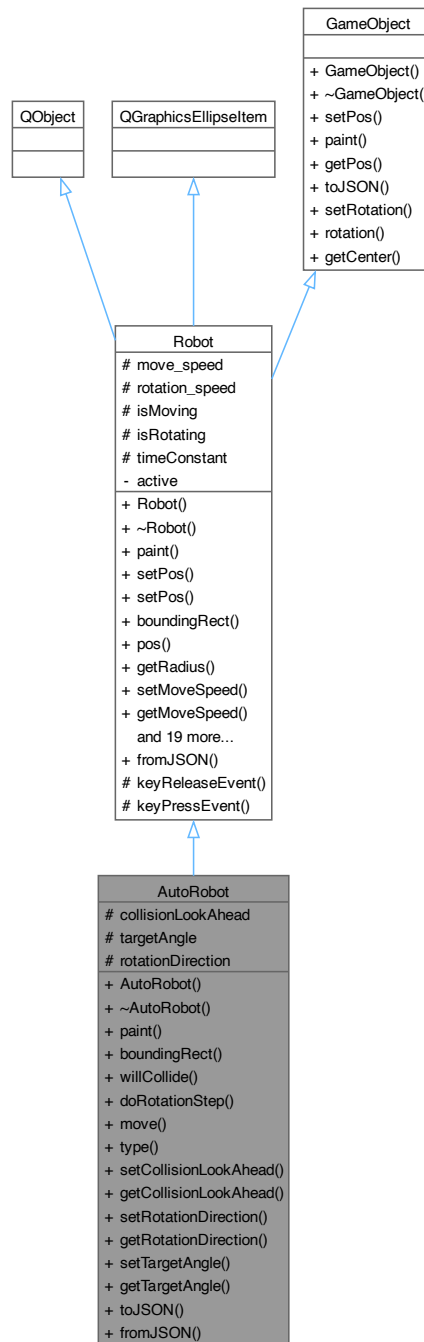
Class Documentation

6.1 AutoRobot Class Reference

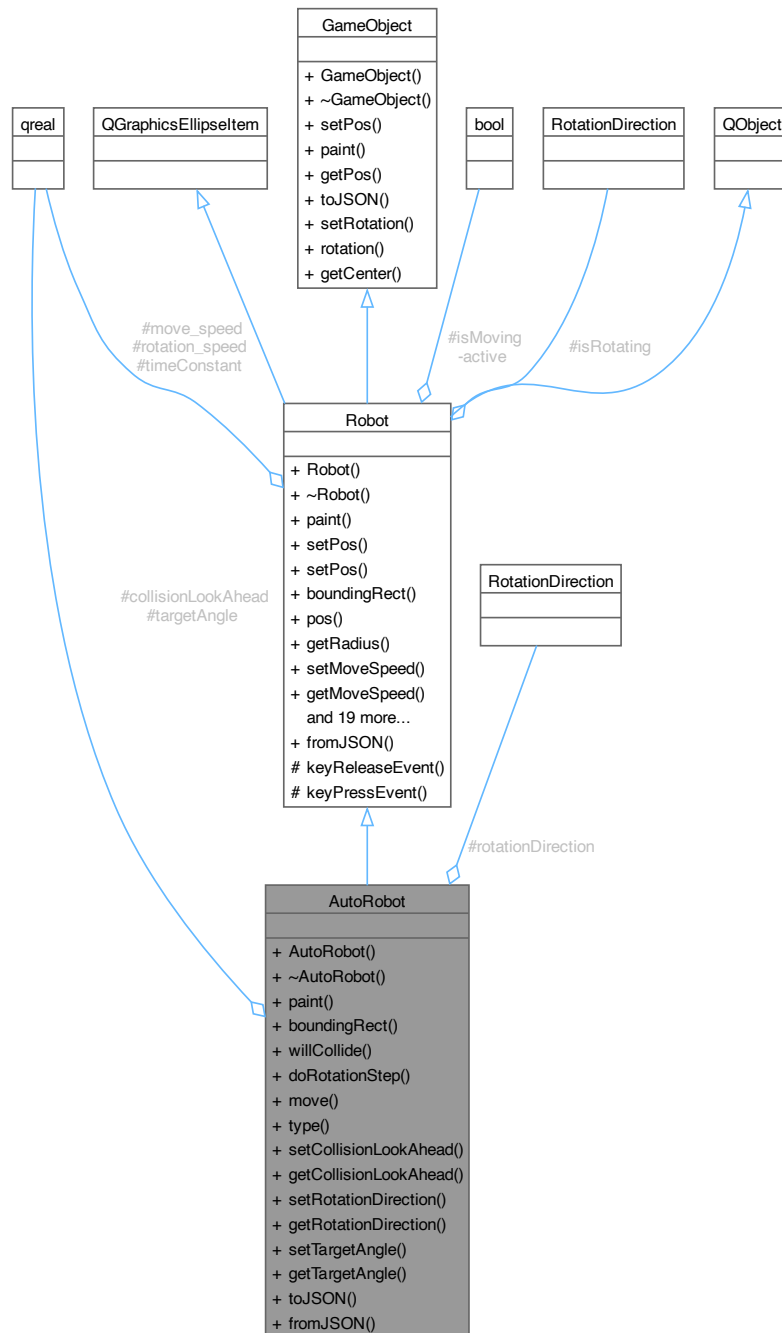
A class to represent an autonomous robot.

```
#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
- *Check if the robot will collide with any object in the scene.*
- void `doRotationStep` (`RotationDirection direction`)
- *Perform a rotation step.*
- bool `move` () override
- *Perform a movement step.*
- int `type` () const override
- *Get the type of the object.*
- void `setCollisionLookAhead` (`qreal lookAhead`)
- *Set the look ahead distance for collision detection.*
- `qreal getCollisionLookAhead` ()
- *Get the look ahead distance for collision detection.*
- void `setRotationDirection` (`RotationDirection direction`)
- *Set the rotation direction of the robot.*
- `RotationDirection getRotationDirection` ()
- *Get the rotation direction of the robot.*
- void `setTargetAngle` (`qreal angle`)
- *Set the target angle of the robot.*
- `qreal getTargetAngle` ()
- *Get the target angle of the robot.*
- `JsonObject toJson` () override
- *Get the JSON representation of the object.*

Public Member Functions inherited from `Robot`

- `Robot` (`QGraphicsItem *parent=nullptr`, `qreal *timeConstant=nullptr`)
- *Default constructor.*
- `~Robot` ()
- void `setPos` (`const QPointF &pos`)
- void `setPos` (`qreal x`, `qreal y`) 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.
- 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.

6.1.1 Detailed Description

A class to represent an autonomous robot.

This class inherits from [Robot](#) and provides functionalities for an autonomous robot.

See also

[Robot](#)

Definition at line 23 of file [autorobot.hpp](#).

6.1.2 Member Enumeration Documentation

6.1.2.1 anonymous enum

anonymous enum

Enumerator

Type	
------	--

Definition at line 27 of file [autorobot.hpp](#).

```
00027 { Type = QGraphicsItem::UserType + 2 };
```

6.1.3 Constructor & Destructor Documentation

6.1.3.1 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.

6.1.3.2 ~AutoRobot()

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

6.1.4 Member Function Documentation

6.1.4.1 boundingRect()

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

Reimplemented from [Robot](#).

6.1.4.2 doRotationStep()

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

Perform a rotation step.

Parameters

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

Returns

void

6.1.4.3 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

6.1.4.4 getCollisionLookAhead()

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

Get the look ahead distance for collision detection.

Returns

qreal The look ahead distance

Definition at line 87 of file [autorobot.hpp](#).

```
00087 { return collisionLookAhead; }
```

6.1.4.5 getRotationDirection()

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

Get the rotation direction of the robot.

Returns

[RotationDirection](#) The rotation direction

Definition at line 100 of file [autorobot.hpp](#).

```
00100 { return rotationDirection; }
```

6.1.4.6 getTargetAngle()

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

Get the target angle of the robot.

Returns

qreal The target angle

Definition at line 113 of file [autorobot.hpp](#).

```
00113 { return targetAngle; }
```

6.1.4.7 move()

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

Perform a movement step.

Returns

bool Whether the movement step was successful

Reimplemented from [Robot](#).

6.1.4.8 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](#).

6.1.4.9 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

Definition at line 81 of file [autorobot.hpp](#).

```
00081 { collisionLookAhead = lookAhead; }
```

6.1.4.10 setRotationDirection()

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

Set the rotation direction of the robot.

Parameters

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

Returns

void

Definition at line 94 of file [autorobot.hpp](#).

```
00094 { rotationDirection = direction; }
```

6.1.4.11 setTargetAngle()

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

Set the target angle of the robot.

Parameters

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

Returns

void

Definition at line 107 of file [autorobot.hpp](#).

```
00107 { targetAngle = angle; }
```

6.1.4.12 toJSON()

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

Get the JSON representation of the object.

Returns

QJsonObject The JSON representation of the object

Reimplemented from [Robot](#).

6.1.4.13 type()

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

Get the type of the object.

Returns

int The type of the object

Definition at line 74 of file [autorobot.hpp](#).

```
00074 { return Type; }
```

6.1.4.14 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](#).

6.1.5 Member Data Documentation

6.1.5.1 collisionLookAhead

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

The look ahead distance for collision detection.

Definition at line 131 of file [autorobot.hpp](#).

6.1.5.2 rotationDirection

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

The rotation direction of the robot.

Definition at line 137 of file [autorobot.hpp](#).

6.1.5.3 targetAngle

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

The target angle of the robot.

Definition at line 134 of file [autorobot.hpp](#).

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

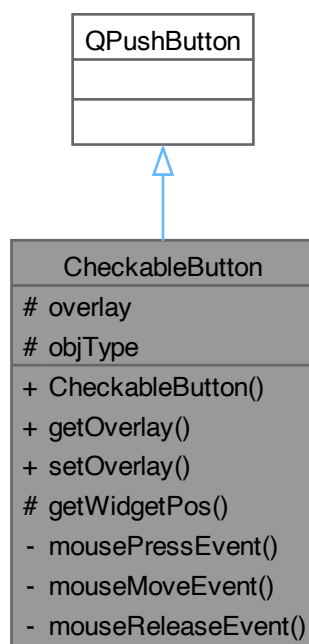
- [autorobot.hpp](#)

6.2 CheckableButton Class Reference

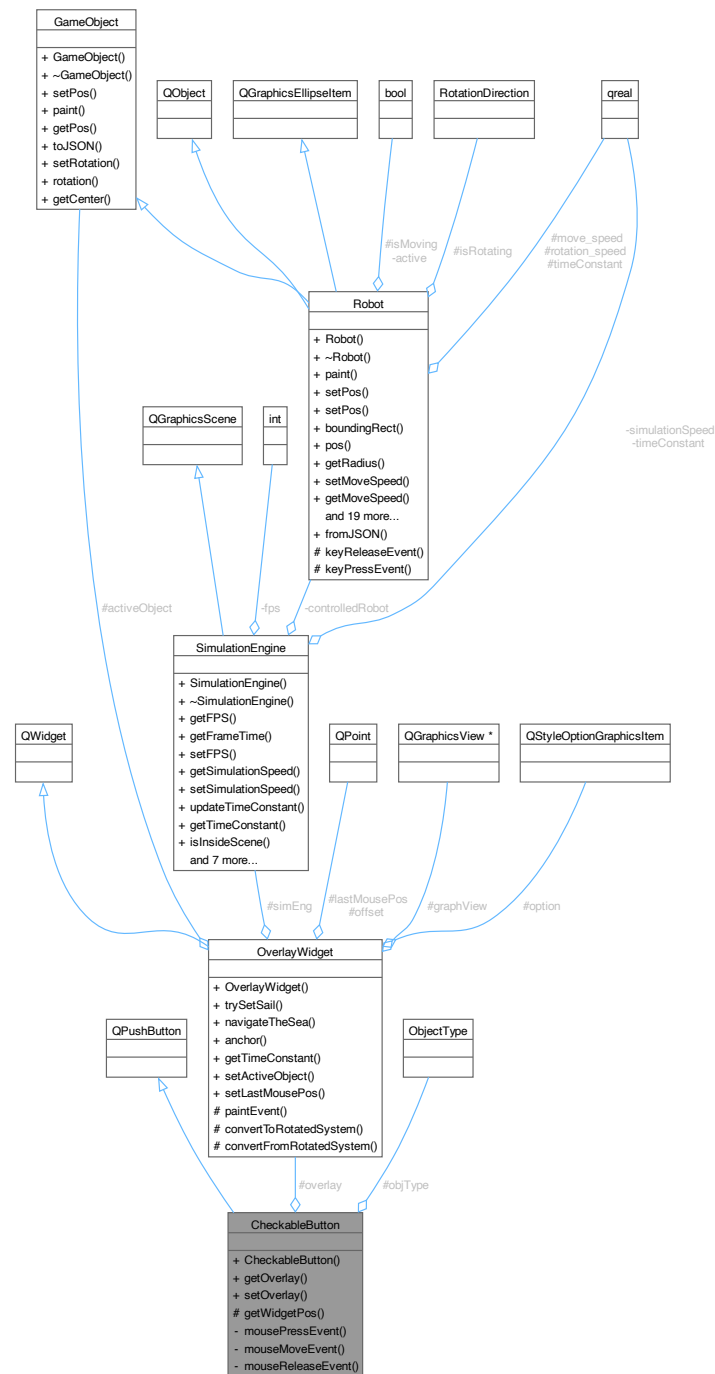
A class to represent a checkable button.

```
#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.

6.2.1 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](#)

Definition at line 24 of file [checkablebutton.hpp](#).

6.2.2 Member Enumeration Documentation

6.2.2.1 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	

Definition at line 31 of file [checkablebutton.hpp](#).

```
00031     {
00032         AUTO,
00033         CONT,
00034         OBST
00035     };
```

6.2.3 Constructor & Destructor Documentation

6.2.3.1 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.

6.2.4 Member Function Documentation

6.2.4.1 getOverlay()

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

Get the overlay widget of the button.

Returns

OverlayWidget* The overlay widget of the button

Definition at line 49 of file [checkablebutton.hpp](#).

```
00049 { return overlay; }
```

6.2.4.2 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.

6.2.4.3 mouseMoveEvent

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

Override the mouseMoveEvent method.

Parameters

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

Returns

void

6.2.4.4 mousePressEvent

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

Override the mousePressEvent method.

Parameters

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

Returns

void

6.2.4.5 mouseReleaseEvent

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

Override the mouseReleaseEvent method.

Parameters

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

Returns

void

6.2.4.6 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

Definition at line 56 of file [checkablebutton.hpp](#).

```
00056 { this->overlay = overlay; }
```

6.2.5 Member Data Documentation

6.2.5.1 objType

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

The type of object that the button represents.

Definition at line 63 of file [checkablebutton.hpp](#).

6.2.5.2 overlay

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

Pointer to the overlay widget.

Definition at line 60 of file [checkablebutton.hpp](#).

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

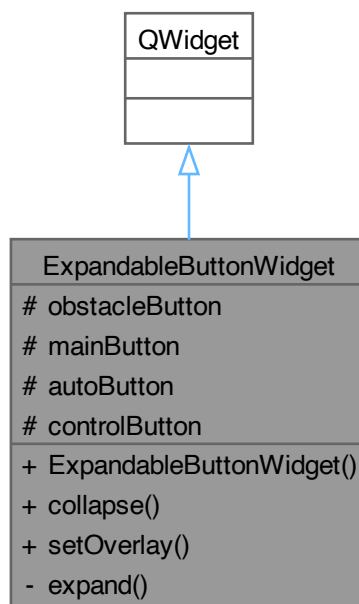
- [checkablebutton.hpp](#)

6.3 ExpandableButtonWidget Class Reference

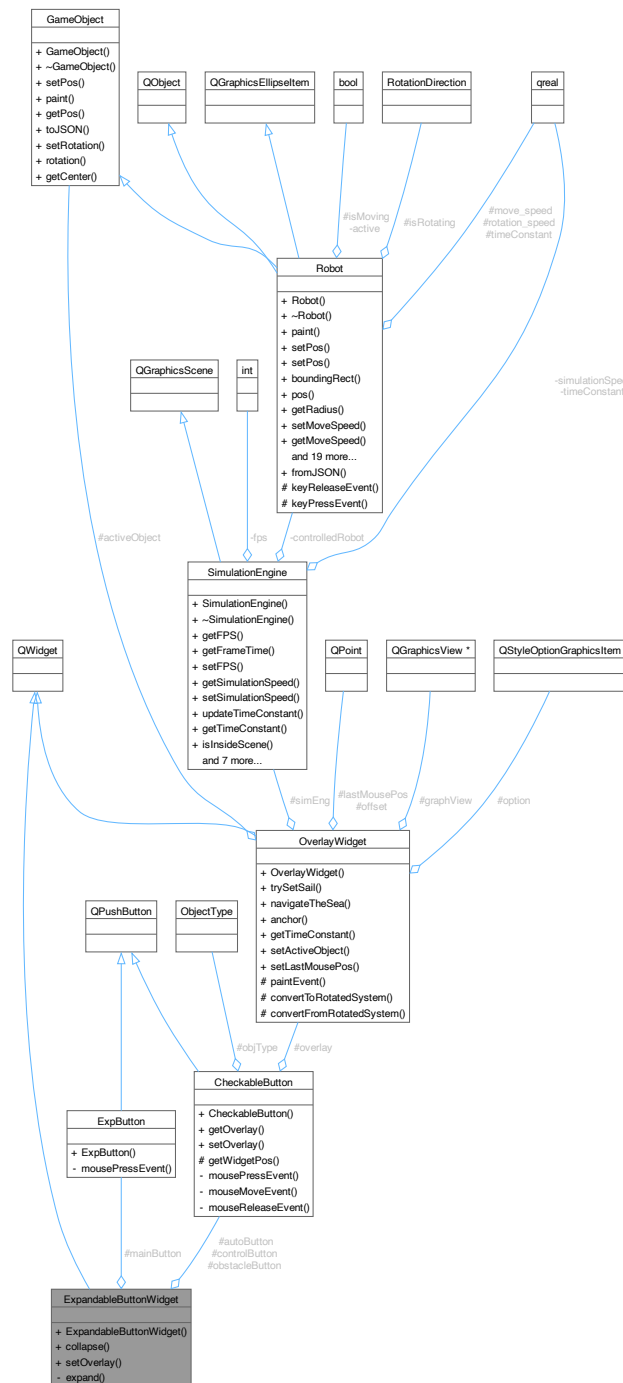
A class to represent an expandable button widget.

```
#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.

6.3.1 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](#)

Definition at line 30 of file [expbuttonwidget.hpp](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 ExpandableButtonWidget()

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

Construct a new Expandable Button Widget object.

Parameters

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

6.3.3 Member Function Documentation

6.3.3.1 collapse()

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

Get the obstacle button.

Returns

CheckableButton* The obstacle button.

6.3.3.2 expand

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

Slot to handle the main button press event.

Returns

void

6.3.3.3 setOverlay()

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

Get the obstacle button.

Returns

CheckableButton* The obstacle button.

6.3.4 Member Data Documentation**6.3.4.1 autoButton**

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

Reference to the auto button.

Definition at line 60 of file [expbuttonwidget.hpp](#).

6.3.4.2 controlButton

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

Reference to the control button.

Definition at line 63 of file [expbuttonwidget.hpp](#).

6.3.4.3 mainButton

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

Reference to the main button.

Definition at line 57 of file [expbuttonwidget.hpp](#).

6.3.4.4 obstacleButton

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

Reference to the obstacle button.

Definition at line 54 of file [expbuttonwidget.hpp](#).

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

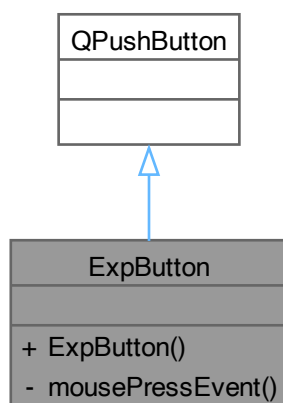
- [expbuttonwidget.hpp](#)

6.4 ExpButton Class Reference

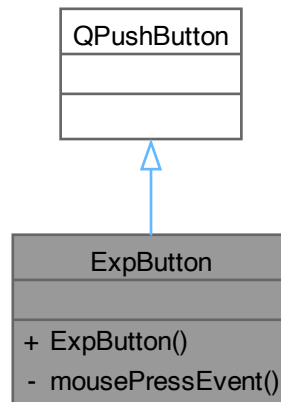
A class for expandable buttons.

```
#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.

6.4.1 Detailed Description

A class for expandable buttons.

This class inherits from [QPushButton](#) and emits a signal when pressed.

See also

[QPushButton](#)

Definition at line 21 of file [expbutton.hpp](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 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 .

6.4.3 Member Function Documentation

6.4.3.1 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

6.4.3.2 pressed

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

Signal emitted when the button is pressed.

Returns

void

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

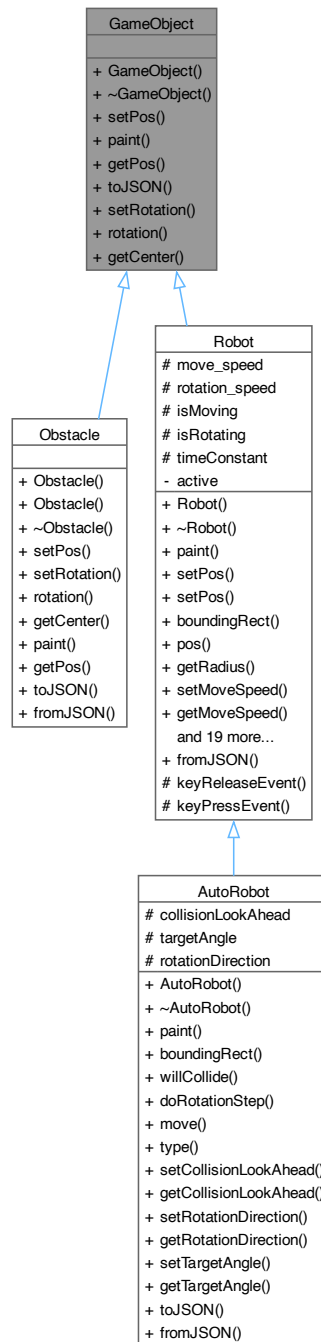
- [expbutton.hpp](#)

6.5 GameObject Class Reference

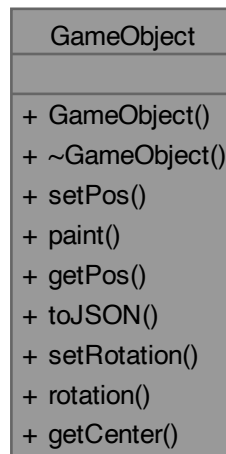
A class to represent a game object in the simulation.

```
#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.

6.5.1 Detailed Description

A class to represent a game object in the simulation.

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

Definition at line 20 of file [gameobject.hpp](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 GameObject()

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

6.5.2.2 ~GameObject()

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

6.5.3 Member Function Documentation

6.5.3.1 getCenter()

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

Get the center of the game object.

Returns

QPointF

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

6.5.3.2 getPos()

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

Get the position of the game object.

Returns

QPointF

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

6.5.3.3 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](#).

6.5.3.4 rotation()

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

Get the rotation of the game object.

Returns

qreal

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

6.5.3.5 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](#).

6.5.3.6 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](#).

6.5.3.7 toJSON()

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

Convert the game object to a JSON object.

Returns

QJsonObject

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

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

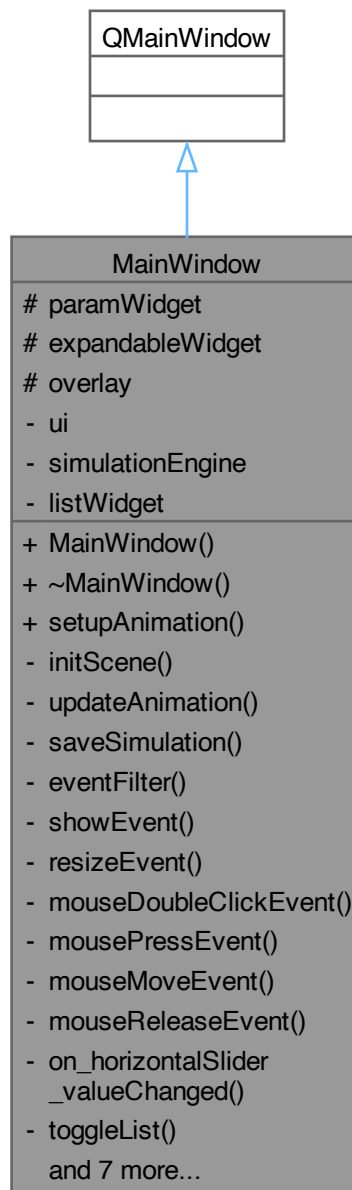
- [gameobject.hpp](#)

6.6 MainWindow Class Reference

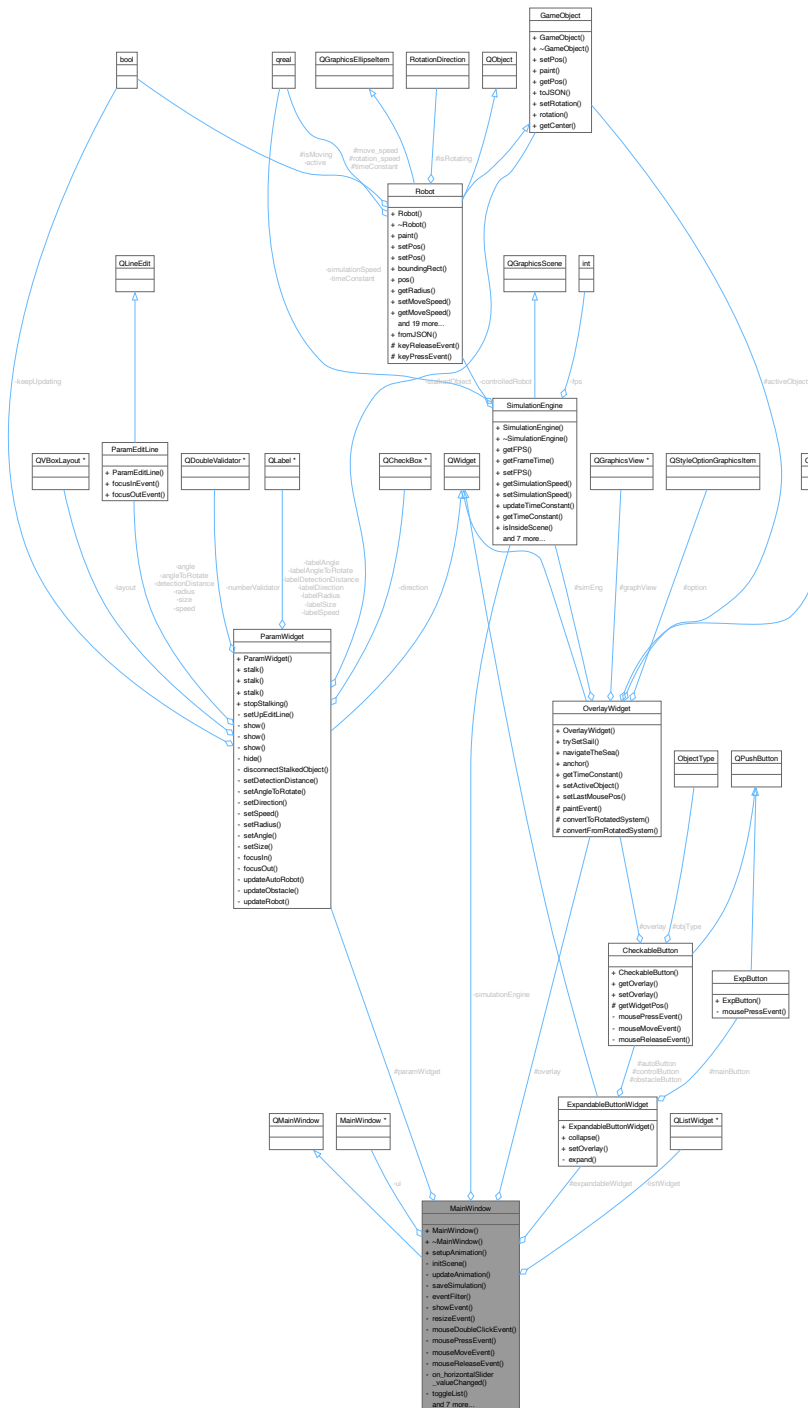
A class to represent the main window of the application.

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

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- `MainWindow (QWidget *parent=nullptr)`
- `~MainWindow ()`
- `void setupAnimation ()`

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.

6.6.1 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](#)

Definition at line 43 of file [mainwindow.h](#).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 MainWindow()

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

6.6.2.2 ~MainWindow()

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

6.6.3 Member Function Documentation

6.6.3.1 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

6.6.3.2 goLeft

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

Slot to handle rotate anticlockwise button click event.

Returns

void

6.6.3.3 goRight

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

Slot to handle rotate clockwise button click event.

Returns

void

6.6.3.4 goStraight

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

Slot to handle move forward button click event.

Returns

void

6.6.3.5 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

6.6.3.6 initScene()

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

6.6.3.7 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

6.6.3.8 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

6.6.3.9 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

6.6.3.10 mousePressEvent

```
void MainWindow::mousePressEvent (  
    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

6.6.3.11 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

6.6.3.12 on_pushButton_clicked

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

Slot to handle clear button click event.

Returns

void

6.6.3.13 `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

6.6.3.14 `saveSimulation`

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

Slot to handle the save button click event.

Returns

void

6.6.3.15 `setupAnimation()`

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

6.6.3.16 `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

6.6.3.17 `stopMoving`

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

Slot to handle stop moving button click event.

Returns

void

6.6.3.18 stopRotating

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

Slot to handle stop rotating button click event.

Returns

void

6.6.3.19 toggleList

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

Slot to handle toggling the list.

Returns

void

6.6.3.20 updateAnimation()

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

6.6.4 Member Data Documentation**6.6.4.1 expandableWidget**

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

The expandable button widget.

Definition at line 69 of file [mainwindow.h](#).

6.6.4.2 listWidget

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

The list widget.

Definition at line 59 of file [mainwindow.h](#).

6.6.4.3 overlay

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

The overlay widget.

Definition at line 72 of file [mainwindow.h](#).

6.6.4.4 paramWidget

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

The param widget.

Definition at line 66 of file [mainwindow.h](#).

6.6.4.5 simulationEngine

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

The simulation engine.

Definition at line 56 of file [mainwindow.h](#).

6.6.4.6 ui

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

The UI object.

Definition at line 53 of file [mainwindow.h](#).

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

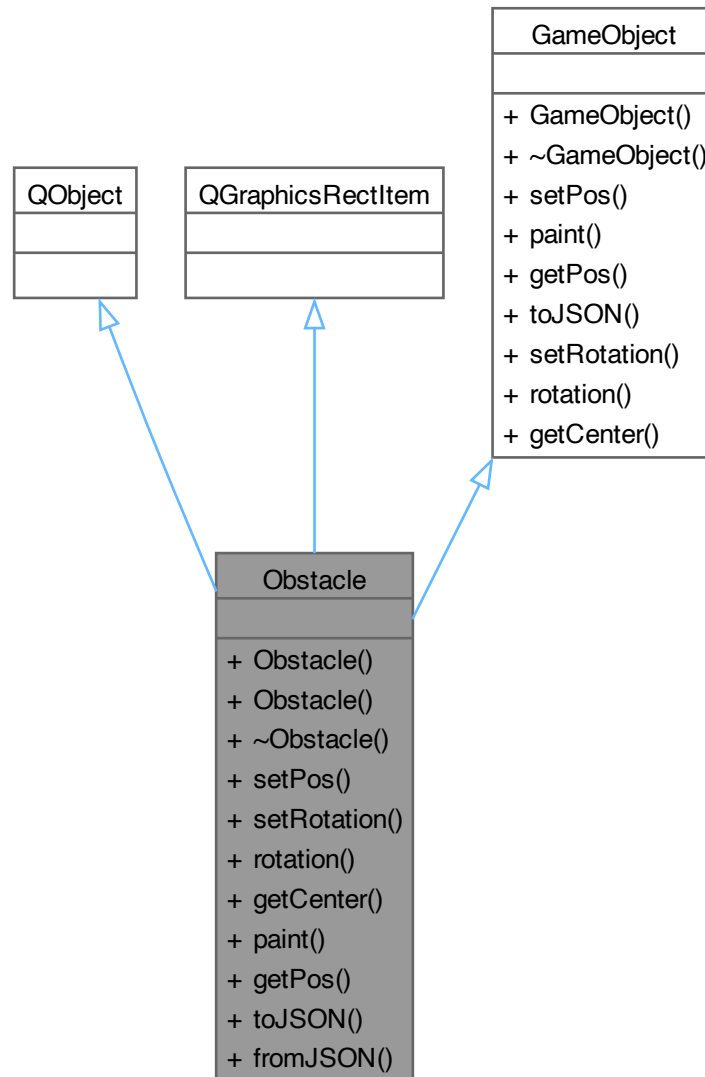
- [mainwindow.h](#)

6.7 Obstacle Class Reference

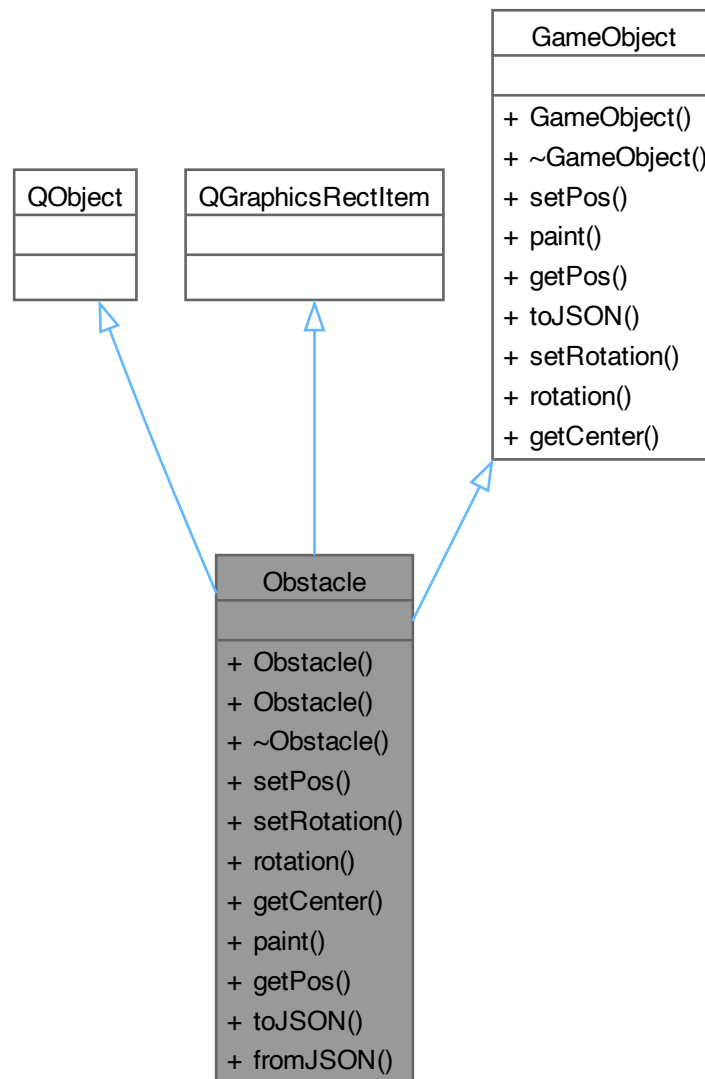
A class to represent an obstacle.

```
#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.

6.7.1 Detailed Description

A class to represent an obstacle.

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

Definition at line 23 of file [obstacle.hpp](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Obstacle() [1/2]

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

Default constructor.

Parameters

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

Returns

void

6.7.2.2 Obstacle() [2/2]

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

Copy constructor.

Parameters

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

Returns

void

Definition at line 39 of file [obstacle.hpp](#).
00040 : QGraphicsRectItem() {}

6.7.2.3 ~Obstacle()

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

Destructor.

6.7.3 Member Function Documentation**6.7.3.1 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.

6.7.3.2 getCenter()

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

Get the center of the obstacle.

Returns

The center of the obstacle as a QPointF.

Implements [GameObject](#).

Definition at line 72 of file [obstacle.hpp](#).

```
00072 { return boundingRect().center(); }
```

6.7.3.3 getPos()

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

Get the position of the obstacle.

Returns

The position of the obstacle as a QPointF object.

Implements [GameObject](#).

6.7.3.4 obstacleSepuku

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

Signal emitted when the obstacle is removed.

Returns

void

6.7.3.5 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](#).

6.7.3.6 paramsUpdated

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

Signal emitted when the parameters of the obstacle are updated.

Returns

void

6.7.3.7 rotation()

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

Get the rotation of the obstacle.

Returns

The rotation of the obstacle as a qreal.

Implements [GameObject](#).

Definition at line 66 of file [obstacle.hpp](#).

```
00066 { return QGraphicsRectItem::rotation(); }
```

6.7.3.8 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](#).

6.7.3.9 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](#).

6.7.3.10 toJSON()

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

Convert the obstacle to a JSON object.

Returns

The obstacle as a QJsonObject.

Implements [GameObject](#).

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

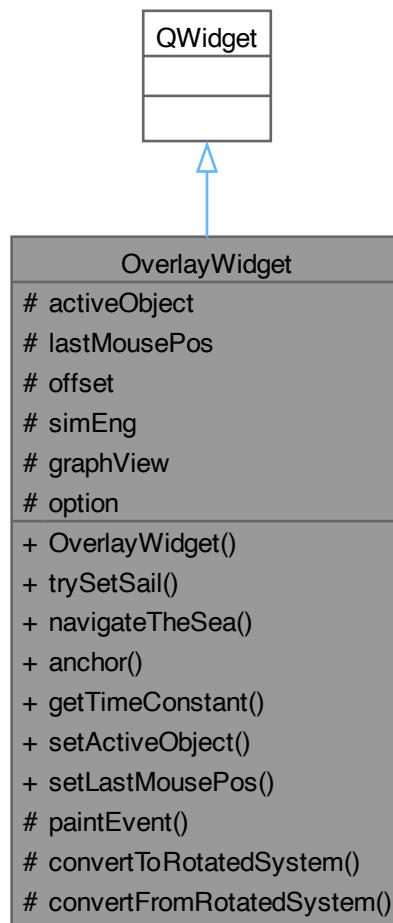
- [obstacle.hpp](#)

6.8 OverlayWidget Class Reference

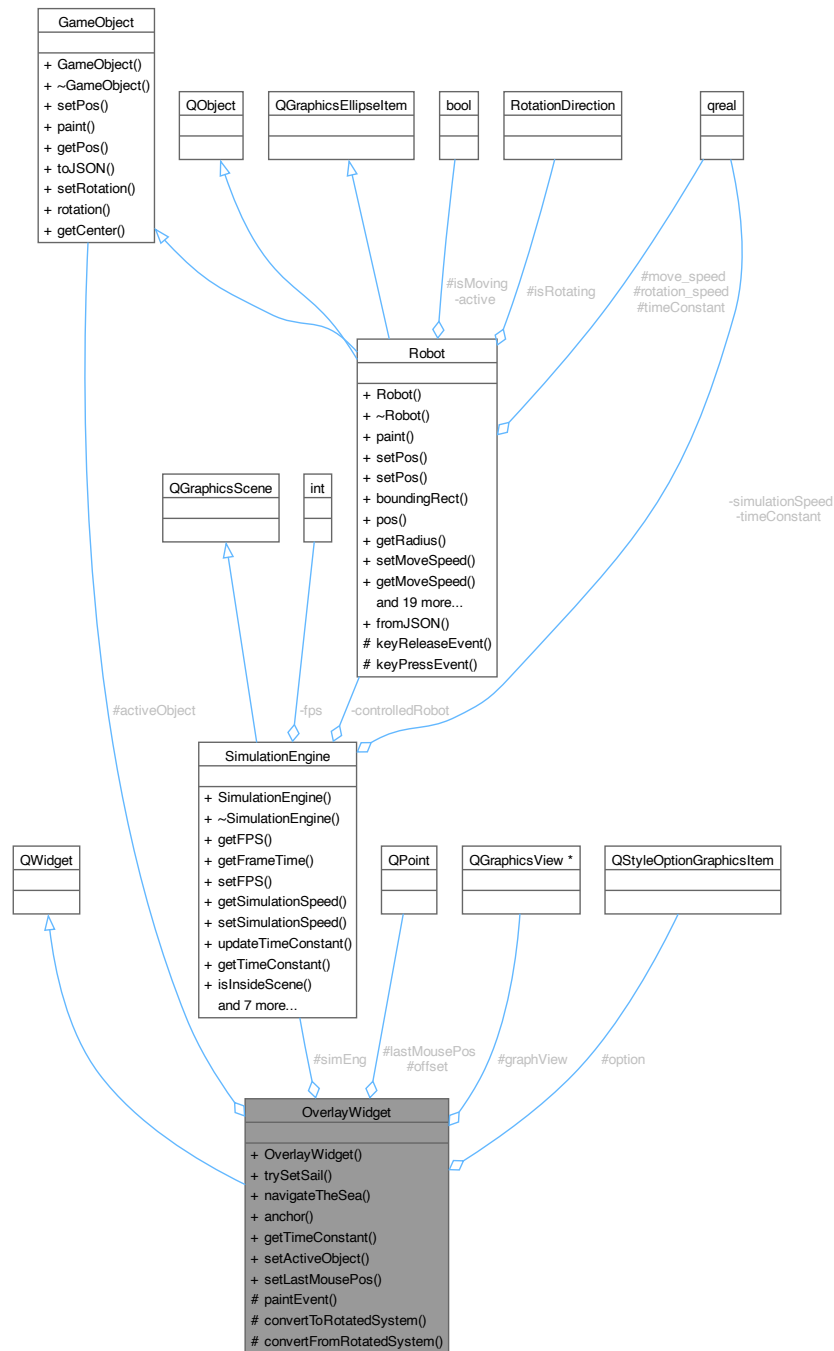
A class to represent an overlay widget.

```
#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.

6.8.1 Detailed Description

A class to represent an overlay widget.

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

See also

[QWidget](#)

Definition at line 27 of file [overlaywidget.hpp](#).

6.8.2 Constructor & Destructor Documentation

6.8.2.1 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.

6.8.3 Member Function Documentation

6.8.3.1 anchor()

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

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

Returns

void

6.8.3.2 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.

6.8.3.3 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.

6.8.3.4 getTimeConstant()

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

Get the time constant of the simulation engine.

Returns

qreal* The time constant of the simulation engine.

Definition at line 62 of file [overlaywidget.hpp](#).

```
00062 { return simEng->getTimeConstant(); }
```

6.8.3.5 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

6.8.3.6 paintEvent()

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

Override the mousePressEvent method.

Parameters

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

Returns

void

6.8.3.7 setActiveObject()

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

Get the active object.

Returns

GameObject* The active object.

Definition at line 68 of file [overlaywidget.hpp](#).

```
00068 { activeObject = obj; }
```

6.8.3.8 setLastMousePos()

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

Get the last mouse position.

Returns

QPoint The last mouse position.

Definition at line 74 of file [overlaywidget.hpp](#).

```
00074 { lastMousePos = pos; }
```

6.8.3.9 trySetSail()

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

Try grab the object based on the mouse position.

Parameters

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

Returns

void

6.8.4 Member Data Documentation

6.8.4.1 activeObject

GameObject* OverlayWidget::activeObject [protected]

The active object.

Definition at line 78 of file [overlaywidget.hpp](#).

6.8.4.2 graphView

```
QGraphicsView* OverlayWidget::graphView [protected]
```

The graphics view.

Definition at line 89 of file [overlaywidget.hpp](#).

6.8.4.3 lastMousePos

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

The last mouse position.

Definition at line 81 of file [overlaywidget.hpp](#).

6.8.4.4 offset

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

Definition at line 83 of file [overlaywidget.hpp](#).

6.8.4.5 option

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

The option for the graphics item.

Definition at line 92 of file [overlaywidget.hpp](#).

6.8.4.6 simEng

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

The simulation engine.

Definition at line 86 of file [overlaywidget.hpp](#).

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

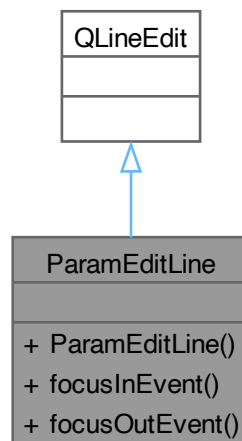
- [overlaywidget.hpp](#)

6.9 ParamEditLine Class Reference

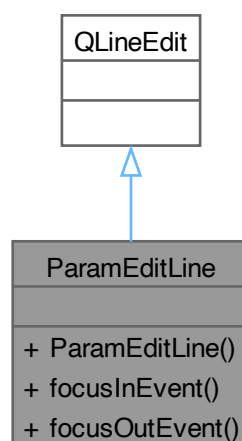
A class to represent a line edit widget for editing parameters.

```
#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.

6.9.1 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](#)

Definition at line 21 of file [parameditline.hpp](#).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 ParamEditLine()

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

Default constructor.

Parameters

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

Definition at line 29 of file [parameditline.hpp](#).

```
00030 : QLineEdit (parent) {}
```

6.9.3 Member Function Documentation

6.9.3.1 focusIn

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

Signal emitted when the line edit widget gains focus.

Returns

void

6.9.3.2 focusInEvent()

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

Overridden focusInEvent method.

Parameters

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

Returns

void

Definition at line 37 of file [parameditline.hpp](#).

```
00037 {
00038     QLineEdit::focusOutEvent(event);
00039     emit focusIn();
00040 }
```

6.9.3.3 focusOut

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

Signal emitted when the line edit widget loses focus.

Returns

void

6.9.3.4 focusOutEvent()

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

Overridden focusOutEvent method.

Parameters

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

Returns

void

Definition at line 47 of file [parameditline.hpp](#).

```
00047                                     {
00048         QLineEdit::focusOutEvent(event);
00049         emit focusOut();
00050     }
```

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

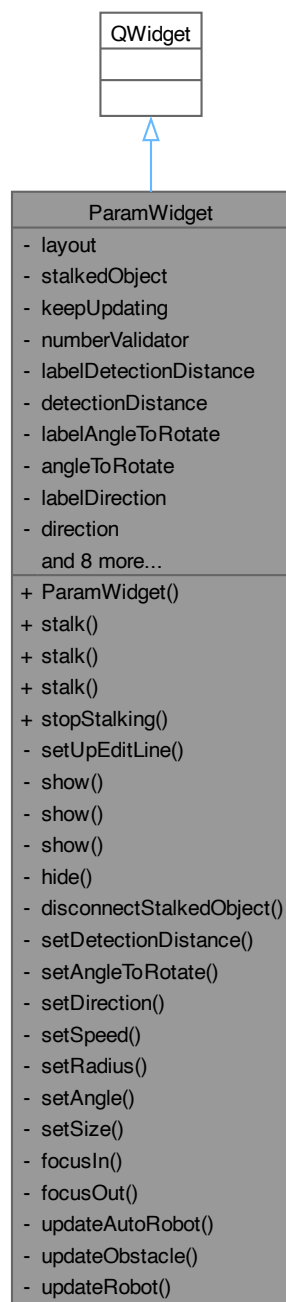
- [parameditline.hpp](#)

6.10 ParamWidget Class Reference

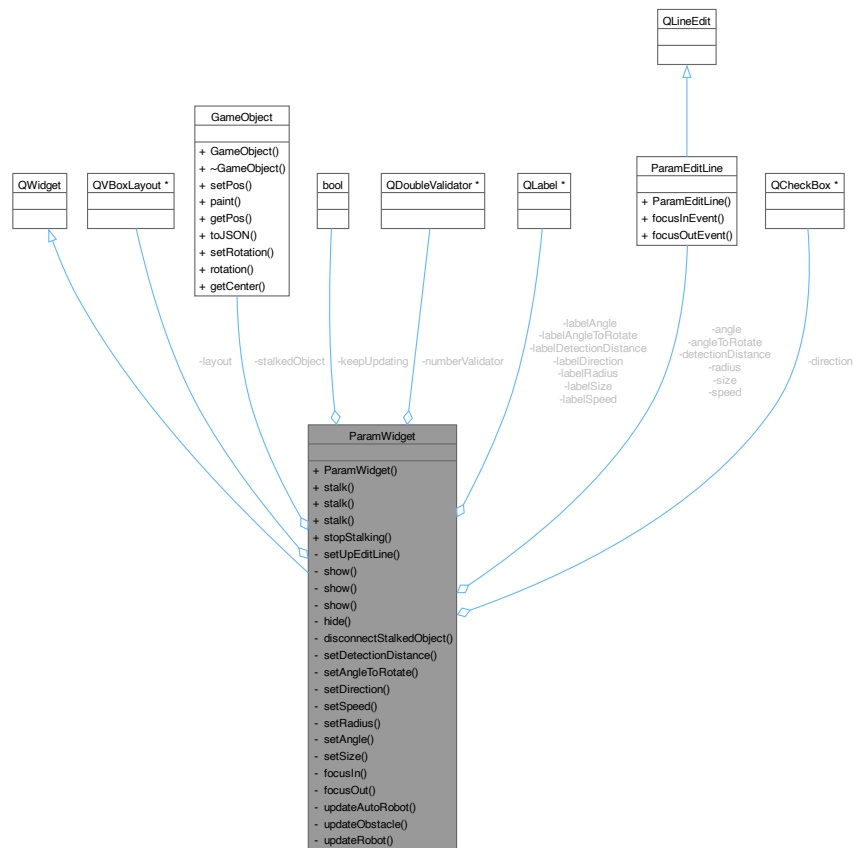
A class to represent a widget for editing parameters of game objects.

```
#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](#)

6.10.1 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](#)

Definition at line 30 of file [paramwidget.hpp](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 ParamWidget()

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

Default constructor.

Parameters

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

6.10.3 Member Function Documentation

6.10.3.1 disconnectStalkedObject()

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

Disconnect the widget from the game object.

Returns

void

6.10.3.2 focusIn

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

Signal to update the parameters of the game object.

Returns

void

Definition at line 184 of file [paramwidget.hpp](#).

```
00184 { keepUpdating = false; }
```

6.10.3.3 focusOut

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

Signal to update the parameters of the game object.

Returns

void

Definition at line 190 of file [paramwidget.hpp](#).

```
00190 { keepUpdating = true; }
```

6.10.3.4 hide()

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

Hide the widget.

Returns

void

6.10.3.5 setAngle

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

Signal to set the angle of the game object.

Returns

void

6.10.3.6 setAngleToRotate

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

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

Returns

void

6.10.3.7 setDetectionDistance

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

Signal to set the detection distance of the game object.

Returns

void

6.10.3.8 setDirection

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

Signal to set the direction of the game object.

Returns

void

6.10.3.9 setRadius

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

Signal to set the radius of the game object.

Returns

void

6.10.3.10 setSize

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

Signal to set the size of the game object.

Returns

void

6.10.3.11 setSpeed

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

Signal to set the speed of the game object.

Returns

void

6.10.3.12 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

6.10.3.13 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

6.10.3.14 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

6.10.3.15 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

6.10.3.16 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

6.10.3.17 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

6.10.3.18 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

6.10.3.19 stopStalking()

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

Stop editing the parameters of the game object.

Returns

void

6.10.3.20 updateAutoRobot

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

Update the parameters of the game object.

Returns

void

6.10.3.21 updateObstacle

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

Update the parameters of the game object.

Returns

void

6.10.3.22 updateRobot

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

Update the parameters of the game object.

Returns

void

6.10.4 Member Data Documentation

6.10.4.1 angle

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

Definition at line 92 of file [paramwidget.hpp](#).

6.10.4.2 angleToRotate

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

Definition at line 84 of file [paramwidget.hpp](#).

6.10.4.3 detectionDistance

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

Definition at line 82 of file [paramwidget.hpp](#).

6.10.4.4 direction

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

Definition at line 86 of file [paramwidget.hpp](#).

6.10.4.5 keepUpdating

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

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

Definition at line 75 of file [paramwidget.hpp](#).

6.10.4.6 labelAngle

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

Definition at line 91 of file [paramwidget.hpp](#).

6.10.4.7 labelAngleToRotate

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

Definition at line 83 of file [paramwidget.hpp](#).

6.10.4.8 labelDetectionDistance

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

The labels and line edit widgets for editing the parameters.

Definition at line 81 of file [paramwidget.hpp](#).

6.10.4.9 labelDirection

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

Definition at line 85 of file [paramwidget.hpp](#).

6.10.4.10 labelRadius

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

Definition at line 89 of file [paramwidget.hpp](#).

6.10.4.11 labelSize

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

Definition at line 93 of file [paramwidget.hpp](#).

6.10.4.12 labelSpeed

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

Definition at line 87 of file [paramwidget.hpp](#).

6.10.4.13 layout

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

The layout of the widget.

Definition at line 69 of file [paramwidget.hpp](#).

6.10.4.14 numberValidator

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

The validator for the number input.

Definition at line 78 of file [paramwidget.hpp](#).

6.10.4.15 radius

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

Definition at line 90 of file [paramwidget.hpp](#).

6.10.4.16 size

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

Definition at line 94 of file [paramwidget.hpp](#).

6.10.4.17 speed

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

Definition at line 88 of file [paramwidget.hpp](#).

6.10.4.18 stalkedObject

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

The game object whose parameters are being displayed.

Definition at line 72 of file [paramwidget.hpp](#).

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

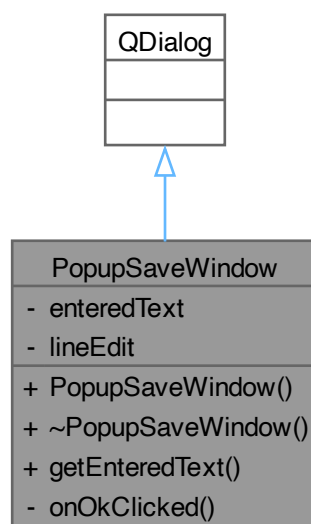
- [paramwidget.hpp](#)

6.11 PopupSaveWindow Class Reference

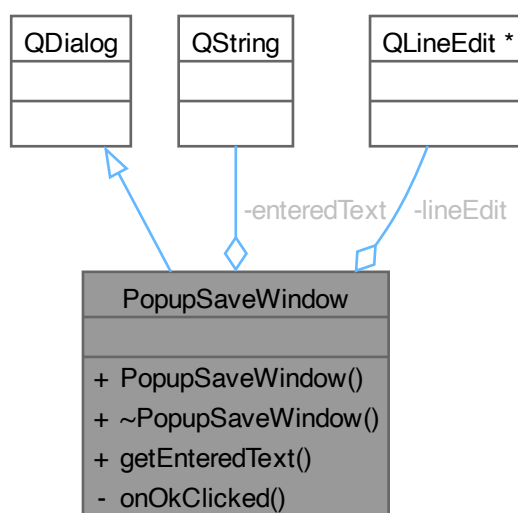
A class to represent a popup save window.

```
#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.

6.11.1 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](#)

Definition at line 25 of file [popsavewindow.h](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 PopupSaveWindow()

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

Construct a new Popup Save Window object.

Parameters

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

6.11.2.2 ~PopupSaveWindow()

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

6.11.3 Member Function Documentation

6.11.3.1 getEnteredText()

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

Get the entered text.

Returns

QString The entered text.

Definition at line 40 of file [popupsavewindow.h](#).

```
00040 { return enteredText; }
```

6.11.3.2 onOkClicked

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

Slot to handle the ok button click event.

Returns

void

6.11.4 Member Data Documentation

6.11.4.1 enteredText

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

The entered text.

Definition at line 44 of file [popupsavewindow.h](#).

6.11.4.2 lineEdit

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

The line edit widget.

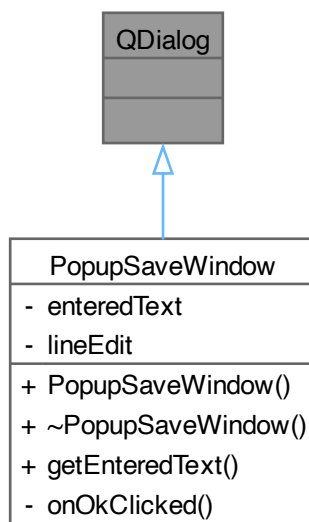
Definition at line 47 of file [popupsavewindow.h](#).

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

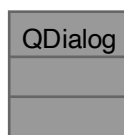
- [popupsavewindow.h](#)

6.12 QDialog Class Reference

Inheritance diagram for QDialog:



Collaboration diagram for QDialog:

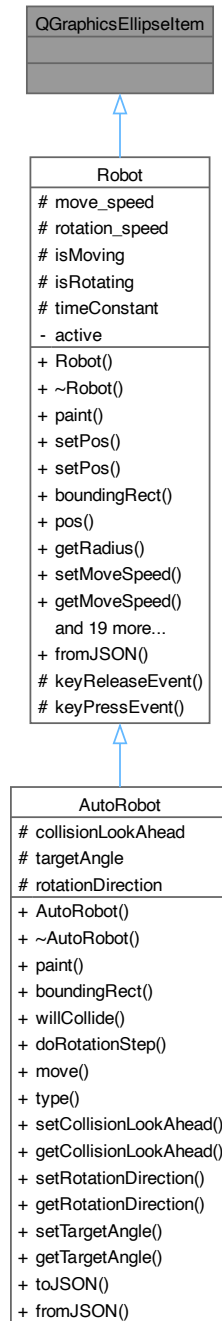


The documentation for this class was generated from the following file:

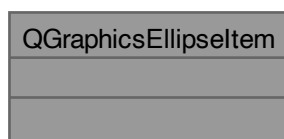
- [popsavewindow.h](#)

6.13 QGraphicsEllipseItem Class Reference

Inheritance diagram for QGraphicsEllipseItem:



Collaboration diagram for QGraphicsEllipseItem:

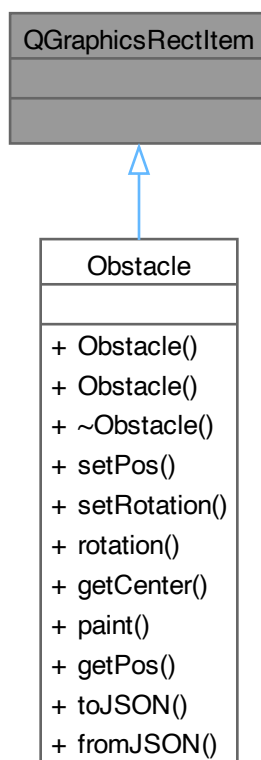


The documentation for this class was generated from the following file:

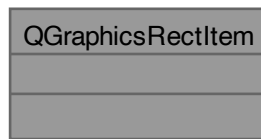
- [robot.hpp](#)

6.14 QGraphicsRectItem Class Reference

Inheritance diagram for QGraphicsRectItem:



Collaboration diagram for QGraphicsRectItem:

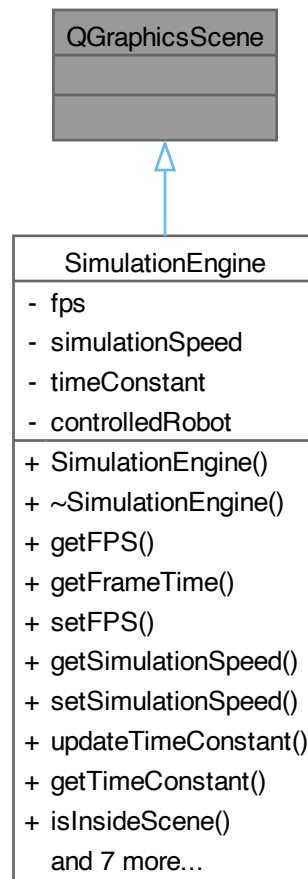


The documentation for this class was generated from the following file:

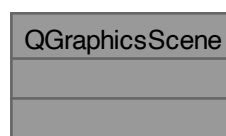
- [obstacle.hpp](#)

6.15 QGraphicsScene Class Reference

Inheritance diagram for QGraphicsScene:



Collaboration diagram for QGraphicsScene:

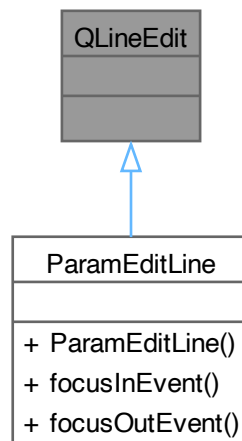


The documentation for this class was generated from the following file:

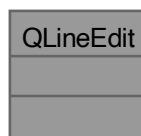
- [simulationengine.hpp](#)

6.16 QLineEdit Class Reference

Inheritance diagram for QLineEdit:



Collaboration diagram for QLineEdit:

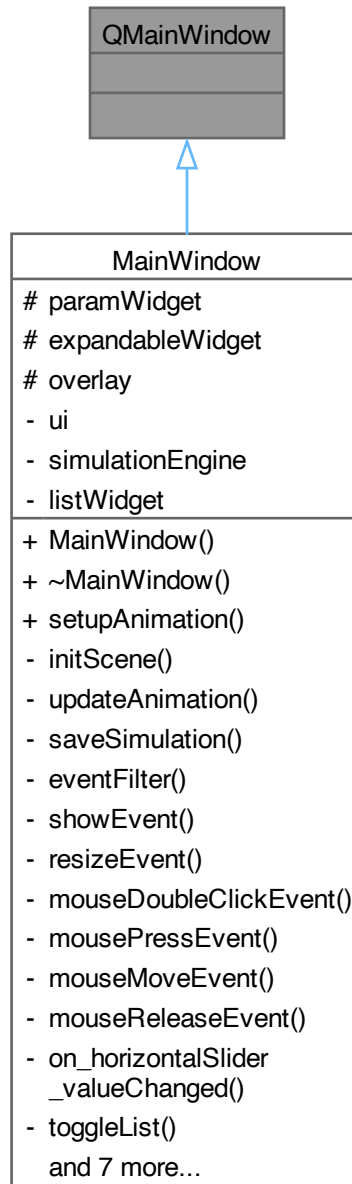


The documentation for this class was generated from the following file:

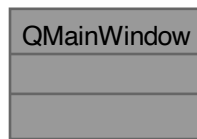
- [parameditline.hpp](#)

6.17 QMainWindow Class Reference

Inheritance diagram for QMainWindow:



Collaboration diagram for QMainWindow:

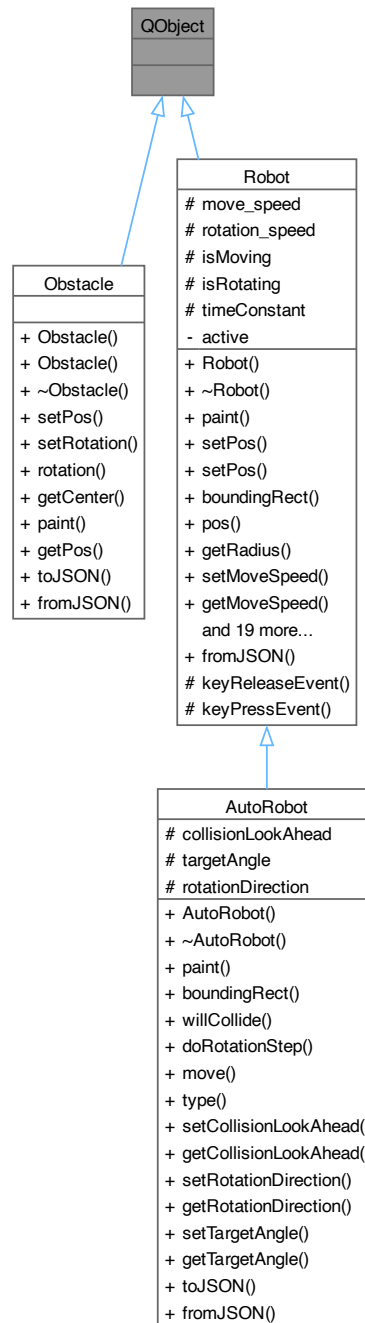


The documentation for this class was generated from the following file:

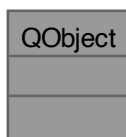
- [mainwindow.h](#)

6.18 QObject Class Reference

Inheritance diagram for QObject:



Collaboration diagram for QObject:

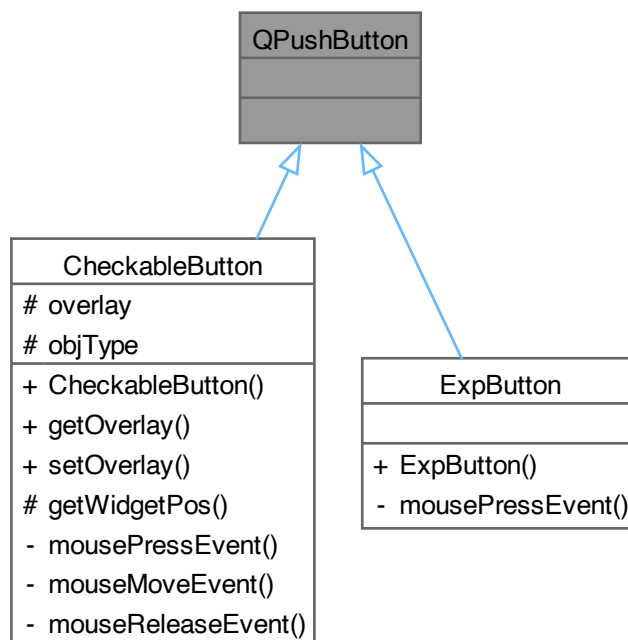


The documentation for this class was generated from the following file:

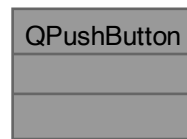
- [obstacle.hpp](#)

6.19 QPushButton Class Reference

Inheritance diagram for QPushButton:



Collaboration diagram for QPushButton:



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

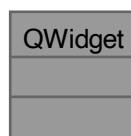
- [checkablebutton.hpp](#)

6.20 QWidget Class Reference

Inheritance diagram for QWidget:



Collaboration diagram for QWidget:



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

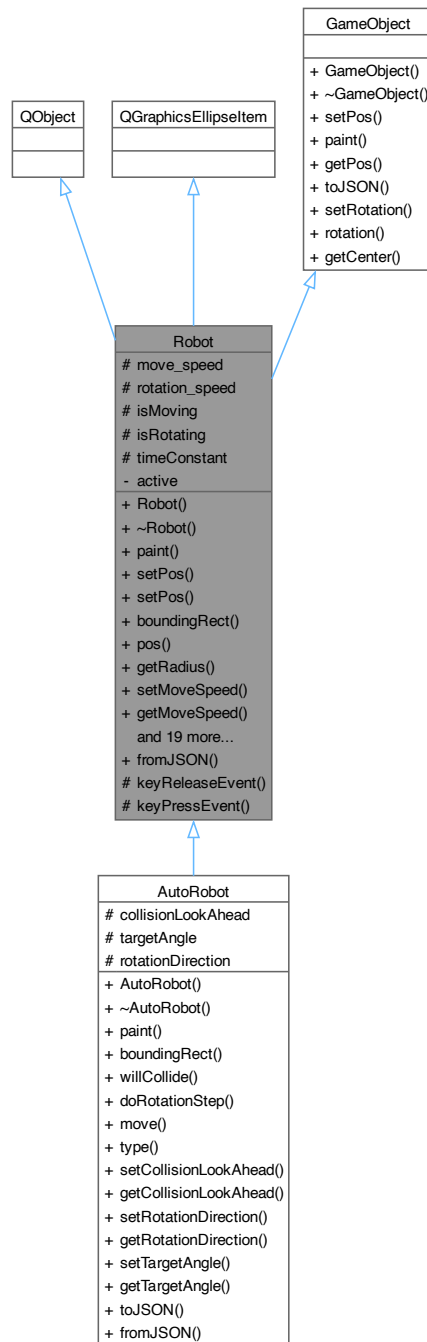
- [expbuttonwidget.hpp](#)

6.21 Robot Class Reference

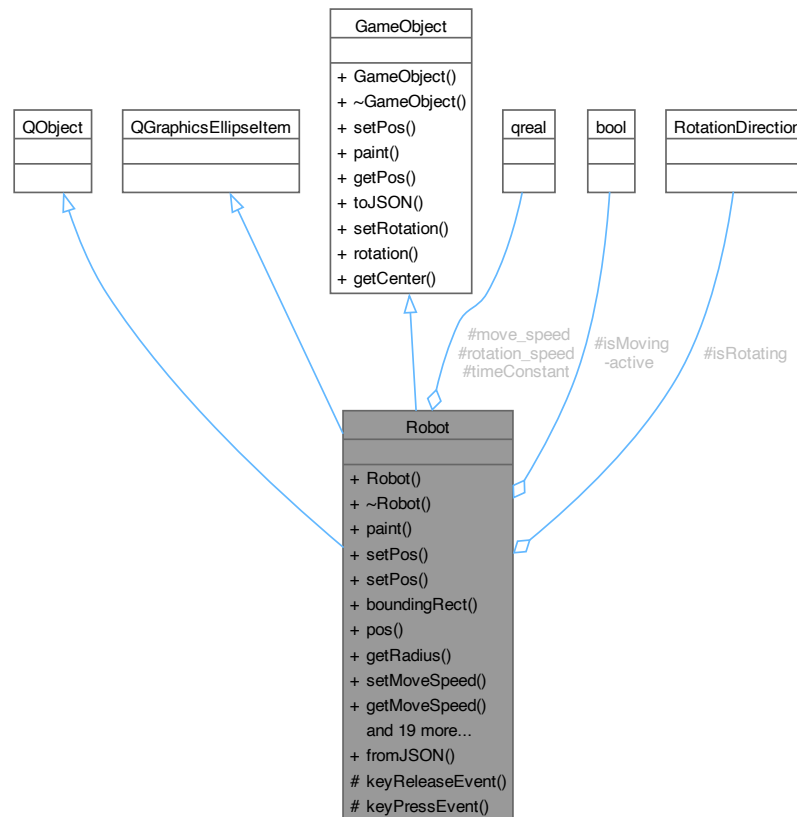
A class to represent a robot in the simulation. By default, the robot is a circle with a line drawn to represent its direction.

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

Public Member Functions

- `Robot` (`QGraphicsItem *parent=nullptr`, `qreal *timeConstant=nullptr`)
- `~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.

6.21.1 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.

Definition at line 27 of file [robot.hpp](#).

6.21.2 Member Enumeration Documentation

6.21.2.1 anonymous enum

anonymous enum

Enumerator

Type	
------	--

Definition at line 40 of file [robot.hpp](#).

```
00040 { Type = QGraphicsItem::UserType + 1 };
```

6.21.2.2 RotationDirection

```
enum Robot::RotationDirection
```

Enum to represent the direction of rotation of the robot.

Enumerator

Left	
None	
Right	

Definition at line 34 of file [robot.hpp](#).

```
00034 {
00035     Left = -1, // Counter-clockwise
00036     None = 0, // No rotation
00037     Right = 1 // Clockwise
00038 };
```

6.21.3 Constructor & Destructor Documentation

6.21.3.1 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.

6.21.3.2 ~Robot()

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

6.21.4 Member Function Documentation

6.21.4.1 boundingRect()

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

Reimplemented in [AutoRobot](#).

6.21.4.2 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*

6.21.4.3 getAngle()

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

Get the angle of the robot.

Returns

qreal

Definition at line 190 of file [robot.hpp](#).

```
00190 { return rotation(); }
```

6.21.4.4 getCenter()

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

Get the center of the robot.

Returns

QPointF

Implements [GameObject](#).

Definition at line 203 of file [robot.hpp](#).

```
00203 { return boundingRect().center(); }
```

6.21.4.5 `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

6.21.4.6 `getMoveSpeed()`

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

Get the move speed of the robot.

Returns

`qreal`

6.21.4.7 `getPos()`

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

Get the position of the robot.

Returns

`QPointF`

Implements [GameObject](#).

6.21.4.8 `getRadius()`

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

6.21.4.9 `getRotationSpeed()`

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

Get the rotation speed of the robot.

Returns

`qreal`

6.21.4.10 isActive()

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

Check if the robot is active.

Returns

bool

Definition at line 184 of file [robot.hpp](#).

```
00184 { return active; }
```

6.21.4.11 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

6.21.4.12 keyReleaseEvent()

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

The radius of the robot.

6.21.4.13 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](#).

6.21.4.14 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](#).

6.21.4.15 paramsUpdated

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

Signal emitted when the parameters of the robot are updated.

Returns

void

6.21.4.16 pos()

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

Override pos to adjust to center-based positioning

6.21.4.17 robotSepuku

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

Signal emitted when the robot is removed.

Returns

void

6.21.4.18 rotation()

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

Get the time constant of the simulation.

Returns

qreal

Implements [GameObject](#).

Definition at line 209 of file [robot.hpp](#).

```
00209         {
00210             return QGraphicsEllipseItem::rotation();
00211         }
```

6.21.4.19 setMoveSpeed()

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

Set the move speed of the robot.

Parameters

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

6.21.4.20 setPos() [1/2]

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

Override setPos to adjust to center-based positioning

6.21.4.21 setPos() [2/2]

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

Overload setPos to accept x and y coordinates

Implements [GameObject](#).

6.21.4.22 setRadius()

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

Set the angle of the robot.

Parameters

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

Returns

void

6.21.4.23 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](#).

Definition at line 218 of file [robot.hpp](#).

```
00218      {  
00219          QGraphicsEllipseItem::setRotation(angle);  
00220      }
```

6.21.4.24 setRotationSpeed()

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

Set the rotation speed of the robot.

Parameters

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

6.21.4.25 startMoving()

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

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

6.21.4.26 startRotating()

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

Start rotating the robot in the given direction.

Parameters

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

6.21.4.27 stopMoving()

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

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

6.21.4.28 stopRotating()

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

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

6.21.4.29 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

Definition at line 175 of file [robot.hpp](#).

```
00175 {
00176     active = !active;
00177     active ? setBrush(QBrush(Qt::lightGray)) : setBrush(QBrush(Qt::transparent));
00178 }
```

6.21.4.30 toJSON()

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

Convert the robot to a JSON object.

Returns

QJsonObject

Implements [GameObject](#).

Reimplemented in [AutoRobot](#).

6.21.4.31 type()

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

Get the type of the robot.

Returns

int

Definition at line 148 of file [robot.hpp](#).

```
00148 { return Type; }
```

6.21.4.32 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](#).

6.21.5 Member Data Documentation

6.21.5.1 active

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

Flag to indicate if the robot is active.

Definition at line 264 of file [robot.hpp](#).

6.21.5.2 isMoving

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

Flag to indicate if the robot is moving.

Definition at line 243 of file [robot.hpp](#).

6.21.5.3 isRotating

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

Flag to indicate the direction of rotation.

Definition at line 246 of file [robot.hpp](#).

6.21.5.4 move_speed

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

The speed of the robot.

Definition at line 238 of file [robot.hpp](#).

6.21.5.5 rotation_speed

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

The speed of the rotation of the robot.

Definition at line 240 of file [robot.hpp](#).

6.21.5.6 timeConstant

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

The time constant of the simulation.

Definition at line 249 of file [robot.hpp](#).

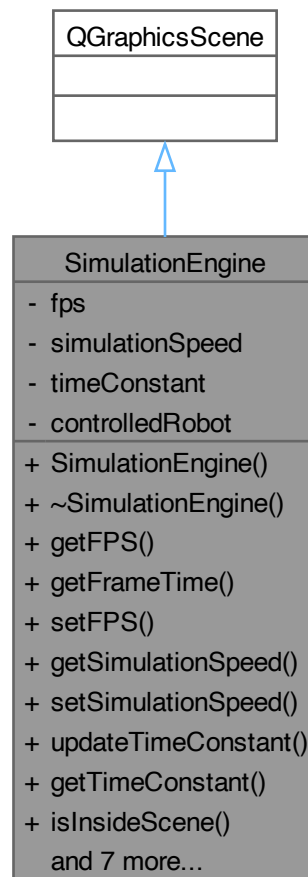
The documentation for this class was generated from the following file:

- [robot.hpp](#)

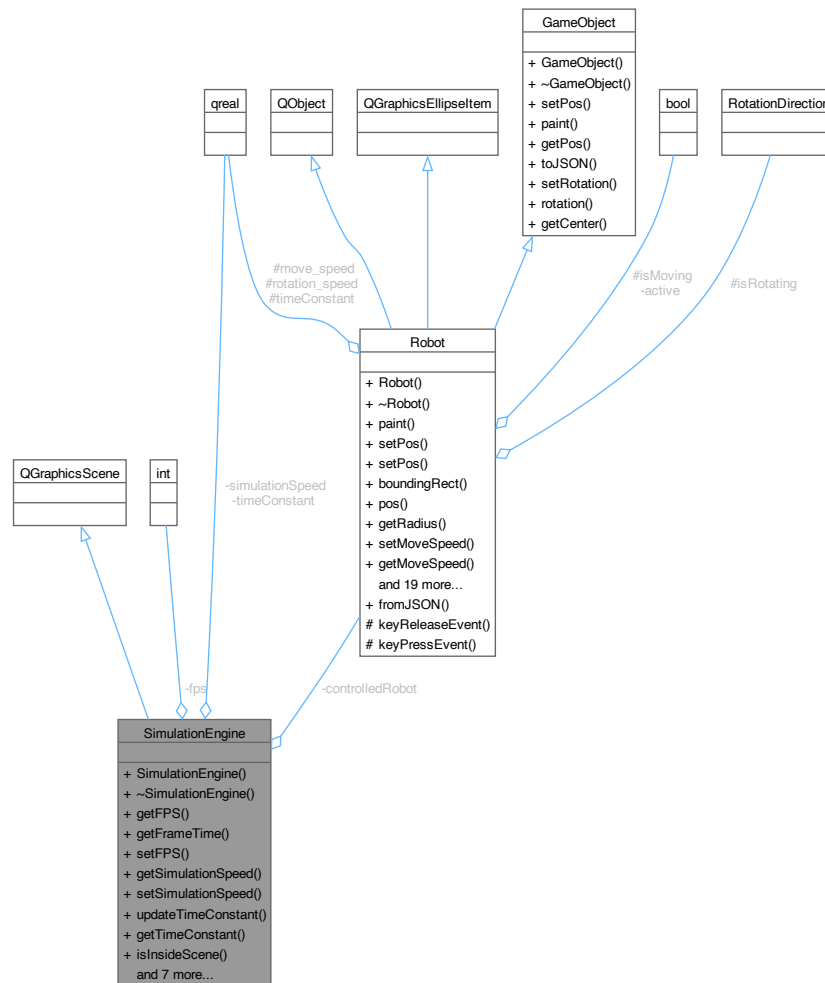
6.22 SimulationEngine Class Reference

```
#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

6.22.1 Detailed Description

Definition at line 19 of file [simulationengine.hpp](#).

6.22.2 Constructor & Destructor Documentation

6.22.2.1 SimulationEngine()

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

6.22.2.2 ~SimulationEngine()

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

6.22.3 Member Function Documentation

6.22.3.1 clearScene()

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

Clear the scene.

6.22.3.2 getControlledRobot()

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

Get the robot that is currently being controlled.

Returns

Robot*

6.22.3.3 getFPS()

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

Simulation Frames-Per-Second getter.

Returns

int

6.22.3.4 getFrameTime()

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

Get the time it takes to render a single frame.

Returns

int

6.22.3.5 getSimulationSpeed()

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

Get the simulation speed.

Returns

qreal

6.22.3.6 getTimeConstant()

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

Get the time constant pointer.

Returns

qreal*

6.22.3.7 isInsideScene()

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

Check if a point is inside the scene.

Parameters

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

Returns

bool

6.22.3.8 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

6.22.3.9 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

6.22.3.10 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

6.22.3.11 setControlledRobot()

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

Set the robot that is currently being controlled.

Parameters

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

Returns

void

6.22.3.12 setFPS()

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

Set the simulation Frames-Per-Second.

Parameters

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

6.22.3.13 setSimulationSpeed()

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

Set the simulation speed.

Parameters

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

Returns

void

6.22.3.14 toJson()

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

Convert the simulation to a JSON object.

Returns

QJsonObject

6.22.3.15 updateTimeConstant()

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

Update the time constant.

Returns

void

6.22.4 Member Data Documentation**6.22.4.1 controlledRobot**

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

The robot that is currently being controlled.

Definition at line 133 of file [simulationengine.hpp](#).

6.22.4.2 fps

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

The frames per second of the simulation engine.

Definition at line 125 of file [simulationengine.hpp](#).

6.22.4.3 simulationSpeed

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

The speed of the simulation engine.

Definition at line 127 of file [simulationengine.hpp](#).

6.22.4.4 timeConstant

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

The time constant of the simulation engine.

Definition at line 130 of file [simulationengine.hpp](#).

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

- [simulationengine.hpp](#)

Chapter 7

File Documentation

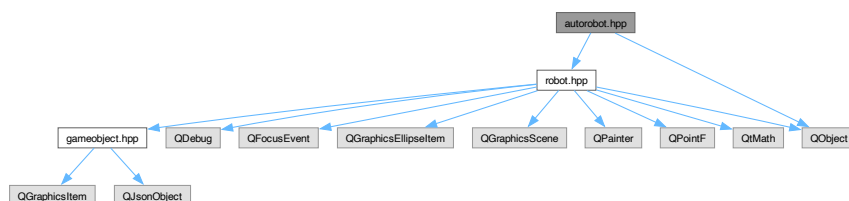
7.1 autorobot.hpp File Reference

This file contains the declaration of the [AutoRobot](#) class.

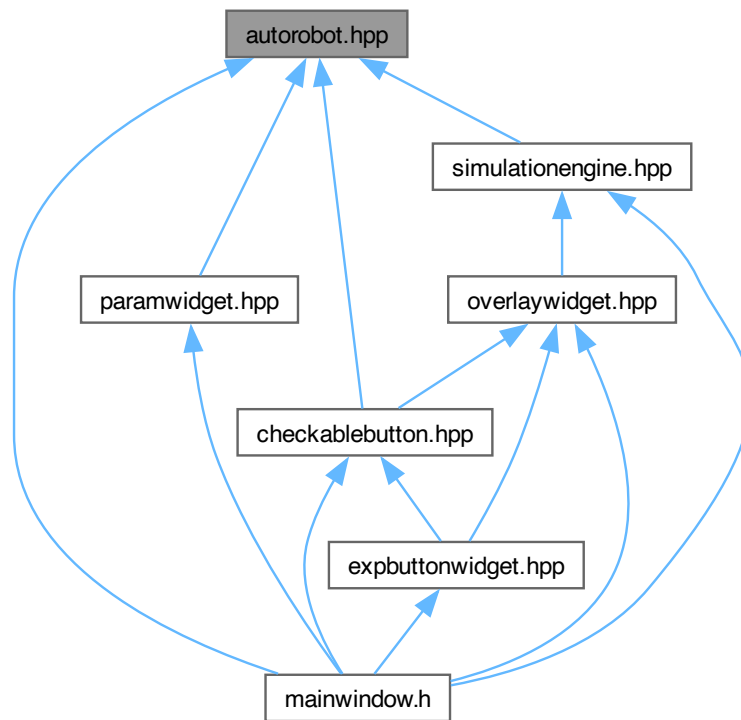
```
#include "robot.hpp"
```

```
#include <QObject>
```

Include dependency graph for autorobot.hpp:



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



Classes

- class [AutoRobot](#)
A class to represent an autonomous robot.

Macros

- #define [SMOOTH_ROTATION_SPEED](#) 0.25

7.1.1 Detailed Description

This file contains the declaration of the [AutoRobot](#) class.

It is a subclass of the [Robot](#) class and represents an autonomous robot.

Authors

Tomáš Hobza, Jakub Všetečka

Date

02.05.2024

Definition in file [autorobot.hpp](#).

7.1.2 Macro Definition Documentation

7.1.2.1 SMOOTH_ROTATION_SPEED

```
#define SMOOTH_ROTATION_SPEED 0.25
```

Definition at line 15 of file [autorobot.hpp](#).

7.2 autorobot.hpp

[Go to the documentation of this file.](#)

```
00001 /**
00002  * @file autorobot.hpp
00003  * @brief This file contains the declaration of the AutoRobot class.
00004  * @details It is a subclass of the Robot class and represents an autonomous robot.
00005  * @authors Tomáš Hobza, Jakub Všecká
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef AUTOROBOT_HPP
00010 #define AUTOROBOT_HPP
00011
00012 #include "robot.hpp"
00013 #include <QObject>
00014
00015 #define SMOOTH_ROTATION_SPEED 0.25
00016
00017 /**
00018  * @class AutoRobot
00019  * @brief A class to represent an autonomous robot.
00020  * @details This class inherits from Robot and provides functionalities for an autonomous robot.
00021  * @see Robot
00022  */
00023 class AutoRobot : public Robot {
00024     // Q_OBJECT
00025
00026     public:
00027         enum { Type = QGraphicsItem::UserType + 2 };
00028
00029         /**
00030          * @brief Constructor for AutoRobot.
00031          * @param parent The parent QGraphicsItem.
00032          * @param size The size of the robot.
00033          * @param collisionLookAhead The distance the robot looks ahead for collisions.
00034          * @param rotationDirection The initial rotation direction of the robot.
00035          * @param moveSpeed The movement speed of the robot.
00036          * @param rotationSpeed The rotation speed of the robot.
00037          * @param timeConstant A pointer to the time constant.
00038          */
00039         AutoRobot(QGraphicsItem *parent = nullptr, qreal size = 50, qreal collisionLookAhead = 10,
00040                 Robot::RotationDirection rotationDirection = Robot::RotationDirection::Right, qreal moveSpeed = 1,
00041                 qreal rotationSpeed = 1, qreal *timeConstant = nullptr);
00042
00043         ~AutoRobot();
00044
00045         /** Override the paint method to draw a line showing the direction of the robot */
00046         void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
00047
00048         /** Override the boundingRect method to adjust the bounding rectangle */
00049         QRectF boundingRect() const override;
00050
00051         /**
00052          * @brief Check if the robot will collide with any object in the scene
00053          * @param directionVector The direction vector of the robot
00054          * @param magnitude The magnitude of the direction vector
00055          * @param allowAnticollision Whether to allow anticollision
00056          * @return bool Whether the robot will collide with any object in the scene
00057          */
00058         bool willCollide(QPointF directionVector, qreal magnitude, bool allowAnticollision) override;
00059
00060         /**
00061          * @brief Perform a rotation step
00062          * @param direction The direction of the rotation
00063          * @return void
00064          */
00065         void doRotationStep(RotationDirection direction);
00066
00067         /**
```

```

00065     * @brief Perform a movement step
00066     * @return bool Whether the movement step was successful
00067     */
00068     bool move() override;
00069
00070     /**
00071     * @brief Get the type of the object
00072     * @return int The type of the object
00073     */
00074     int type() const override { return Type; }
00075
00076     /**
00077     * @brief Set the look ahead distance for collision detection
00078     * @param lookAhead The look ahead distance
00079     * @return void
00080     */
00081     void setCollisionLookAhead(qreal lookAhead) { collisionLookAhead = lookAhead; }
00082
00083     /**
00084     * @brief Get the look ahead distance for collision detection
00085     * @return qreal The look ahead distance
00086     */
00087     qreal getCollisionLookAhead() { return collisionLookAhead; }
00088
00089     /**
00090     * @brief Set the rotation direction of the robot
00091     * @param direction The rotation direction
00092     * @return void
00093     */
00094     void setRotationDirection(RotationDirection direction) { rotationDirection = direction; }
00095
00096     /**
00097     * @brief Get the rotation direction of the robot
00098     * @return RotationDirection The rotation direction
00099     */
00100     RotationDirection getRotationDirection() { return rotationDirection; }
00101
00102     /**
00103     * @brief Set the target angle of the robot
00104     * @param angle The target angle
00105     * @return void
00106     */
00107     void setTargetAngle(qreal angle) { targetAngle = angle; }
00108
00109     /**
00110     * @brief Get the target angle of the robot
00111     * @return qreal The target angle
00112     */
00113     qreal getTargetAngle() { return targetAngle; }
00114
00115     /**
00116     * @brief Get the JSON representation of the object
00117     * @return QJsonObject The JSON representation of the object
00118     */
00119     QJsonObject toJSON() override;
00120
00121     /**
00122     * @brief Create an AutoRobot object from a JSON object
00123     * @param object The JSON object to create the AutoRobot object from
00124     * @param timeConstant The time constant of the robot
00125     * @return AutoRobot* The AutoRobot object created from the JSON object
00126     */
00127     static AutoRobot *fromJSON(const QJsonObject &object, qreal *timeConstant);
00128
00129 protected:
00130     /** @brief The look ahead distance for collision detection */
00131     qreal collisionLookAhead = 0;
00132
00133     /** @brief The target angle of the robot */
00134     qreal targetAngle = 0;
00135
00136     /** @brief The rotation direction of the robot */
00137     Robot::RotationDirection rotationDirection = Robot::RotationDirection::Right;
00138 };
00139
00140 #endif // AUTOROBOT_HPP

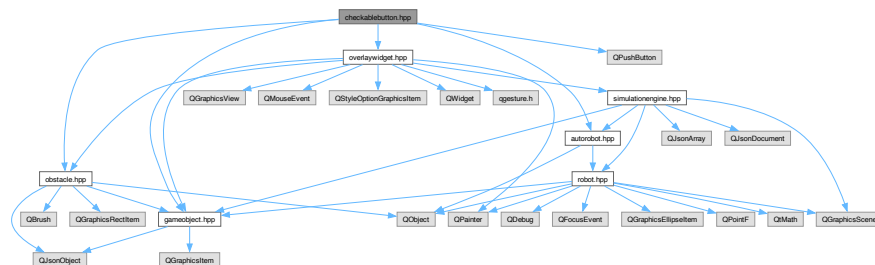
```

7.3 checkablebutton.hpp File Reference

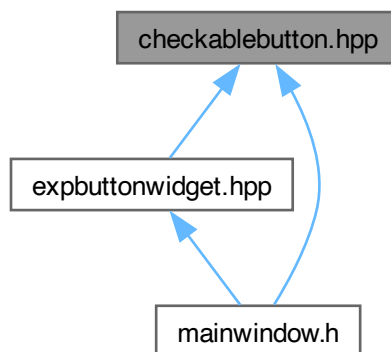
This file contains the declaration of the [CheckableButton](#) class.

```
#include "autorobot.hpp"
#include "gameobject.hpp"
#include "obstacle.hpp"
#include "overlaywidget.hpp"
#include <QPushButton>
```

Include dependency graph for checkablebutton.hpp:



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



Classes

- class `CheckableButton`
A class to represent a checkable button.

7.3.1 Detailed Description

This file contains the declaration of the [CheckableButton](#) class.

It is a subclass of the [QPushButton](#) class and represents a checkable button.

Authors

Tomáš Hobza, Jakub Vřetečka

Date

02.05.2024

Definition in file [checkablebutton.hpp](#).

7.4 checkablebutton.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file checkablebutton.hpp
00003  * @brief This file contains the declaration of the CheckableButton class.
00004  * @details It is a subclass of the QPushButton class and represents a checkable button.
00005  * @authors Tomáš Hobza, Jakub Všeťečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef CHECKABLEBUTTON_HPP
00010 #define CHECKABLEBUTTON_HPP
00011
00012 #include "autorobot.hpp"
00013 #include "gameobject.hpp"
00014 #include "obstacle.hpp"
00015 #include "overlaywidget.hpp"
00016 #include <QPushButton>
00017
00018 /**
00019  * @class CheckableButton
00020  * @brief A class to represent a checkable button.
00021  * @details This class inherits from QPushButton and provides functionalities for a button that can be
00022  * checked and unchecked. It also has an OverlayWidget that is used to draw the object on the grid.
00023  * @see QPushButton
00024  */
00025 class CheckableButton : public QPushButton {
00026 public:
00027     /** * @brief Enum to represent the type of object that the button represents
00028     * AUTO: AutoRobot
00029     * CONT: ControlledRobot
00030     * OBST: Obstacle
00031     */
00032     enum ObjectType {
00033         AUTO,
00034         CONT,
00035         OBST
00036     };
00037
00038     /**
00039     * @brief Constructor for CheckableButton.
00040     * @param text The text to be displayed on the button.
00041     * @param parent The parent QWidget.
00042     * @param type The type of object that the button represents.
00043     */
00044     explicit CheckableButton(const QString &text, QWidget *parent = nullptr, ObjectType type =
00045         ObjectType::OBST);
00046
00047     /**
00048     * @brief Get the overlay widget of the button
00049     * @return OverlayWidget* The overlay widget of the button
00050     */
00051     OverlayWidget *getOverlay() const { return overlay; }
00052
00053     /**
00054     * @brief Set the overlay widget of the button
00055     * @param overlay The overlay widget to set
00056     * @return void
00057     */
00058     void setOverlay(OverlayWidget *overlay) { this->overlay = overlay; }
00059
00060 protected:
00061     /** * @brief Pointer to the overlay widget */
00062     OverlayWidget *overlay;
00063
00064     /** * @brief The type of object that the button represents */
00065     ObjectType objType;
00066
00067     /**
00068     * @brief Get the position of the widget on the grid.
00069     * @param localPos The local position of the mouse.
00070     * @return QPoint The position in the overlay widget.

```

```
00069     */
00070     QPoint getWidgetPos(QPoint localPos);
00071
00072 private slots:
00073
00074     /**
00075      * @brief Override the mousePressEvent method
00076      * @param event The mouse event
00077      * @return void
00078      */
00079     void mousePressEvent(QMouseEvent *event) override;
00080
00081     /**
00082      * @brief Override the mouseMoveEvent method
00083      * @param event The mouse event
00084      * @return void
00085      */
00086     void mouseMoveEvent(QMouseEvent *event) override;
00087
00088     /**
00089      * @brief Override the mouseReleaseEvent method
00090      * @param event The mouse event
00091      * @return void
00092      */
00093     void mouseReleaseEvent(QMouseEvent *event) override;
00094 };
00095
00096 #endif // CHECKABLEBUTTON_HPP
```

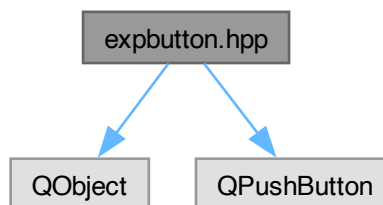
7.5 expbutton.hpp File Reference

This file contains the declaration of the [ExpButton](#) class.

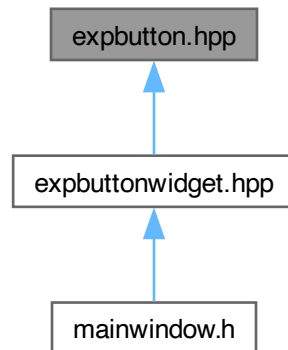
```
#include <QObject>
```

```
#include <QPushButton>
```

Include dependency graph for expbutton.hpp:



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



Classes

- class [ExpButton](#)
A class for expandable buttons.

7.5.1 Detailed Description

This file contains the declaration of the [ExpButton](#) class.

It is a subclass of the [QPushButton](#) class and represents an expandable button.

Authors

Tomáš Hobza, Jakub Vřetečka

Date

02.05.2024

Definition in file [expbutton.hpp](#).

7.6 expbutton.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file expbutton.hpp
00003  * @brief This file contains the declaration of the ExpButton class.
00004  * @details It is a subclass of the QPushButton class and represents an expandable button.
00005  * @authors Tomáš Hobza, Jakub Všecký
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef EXPBUTTON_HPP
00010 #define EXPBUTTON_HPP
00011
00012 #include <QObject>
00013 #include <QPushButton>
00014
00015 /**
00016  * @class ExpButton
00017  * @brief A class for expandable buttons.
00018  * @details This class inherits from QPushButton and emits a signal when pressed.
00019  * @see QPushButton
00020  */
00021 class ExpButton : public QPushButton {
00022     Q_OBJECT
00023
00024     public:
00025         /**
00026          * @brief Constructor for ExpButton.
00027          * @param text The text to be displayed on the button.
00028          * @param parent The parent QWidget.
00029          */
00030         explicit ExpButton(const QString &text, QWidget *parent = nullptr);
00031
00032     signals:
00033         /**
00034          * @brief Signal emitted when the button is pressed.
00035          * @return void
00036          */
00037         void pressed();
00038
00039     private slots:
00040         /**
00041          * @brief Slot to handle the button press event.
00042          * @param event The QMouseEvent that triggered the slot.
00043          * @return void
00044          */
00045         void mousePressEvent(QMouseEvent *event) override;
00046 };
00047
00048 #endif // EXPBUTTON_HPP

```

7.7 expbuttonwidget.hpp File Reference

This file contains the declaration of the [ExpandableButtonWidget](#) class.

```

#include "checkablebutton.hpp"
#include "expbutton.hpp"
#include "overlaywidget.hpp"
#include <QApplication>
#include <QDebug>
#include <QEvent>
#include <QMouseEvent>
#include <QPushButton>
#include <QVBoxLayout>
#include <QWidget>

```


7.8 expbuttonwidget.hpp

[Go to the documentation of this file.](#)

```

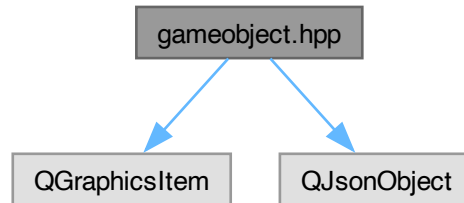
00001 /**
00002  * @file expbuttonwidget.hpp
00003  * @brief This file contains the declaration of the ExpandableButtonWidget class.
00004  * @details It is a subclass of the QWidget class and represents an expandable button widget.
00005  * @authors Tomáš Hobza, Jakub Všetěčka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef EXPANDABLEBUTTONWIDGET_HPP
00010 #define EXPANDABLEBUTTONWIDGET_HPP
00011
00012 // ExpandableButtonWidget.h
00013 #include "checkablebutton.hpp"
00014 #include "expbutton.hpp"
00015 #include "overlaywidget.hpp"
00016 #include <QApplication>
00017 #include <QDebug>
00018 #include <QEvent>
00019 #include <QMouseEvent>
00020 #include <QPushButton>
00021 #include <QVBoxLayout>
00022 #include <QWidget>
00023
00024 /**
00025  * @class ExpandableButtonWidget
00026  * @brief A class to represent an expandable button widget.
00027  * @details This class provides an interface for creating and managing expandable button widgets.
00028  * @see QWidget
00029  */
00030 class ExpandableButtonWidget : public QWidget {
00031     Q_OBJECT
00032
00033 public:
00034     /**
00035      * @brief Construct a new Expandable Button Widget object.
00036      * @param parent The parent widget. Default is nullptr.
00037      */
00038     explicit ExpandableButtonWidget(QWidget *parent = nullptr);
00039
00040     /**
00041      * @brief Get the obstacle button.
00042      * @return CheckableButton* The obstacle button.
00043      */
00044     void collapse();
00045
00046     /**
00047      * @brief Get the obstacle button.
00048      * @return CheckableButton* The obstacle button.
00049      */
00050     void setOverlay(OverlayWidget *overlay);
00051
00052 protected:
00053     /** @brief Reference to the obstacle button.*/
00054     CheckableButton *obstacleButton;
00055
00056     /** @brief Reference to the main button.*/
00057     ExpButton *mainButton;
00058
00059     /** @brief Reference to the auto button.*/
00060     CheckableButton *autoButton;
00061
00062     /** @brief Reference to the control button.*/
00063     CheckableButton *controlButton;
00064
00065 private slots:
00066     /**
00067      * @brief Slot to handle the main button press event.
00068      * @return void
00069      */
00070     void expand();
00071 };
00072
00073 #endif // EXPANDABLEBUTTONWIDGET_HPP

```

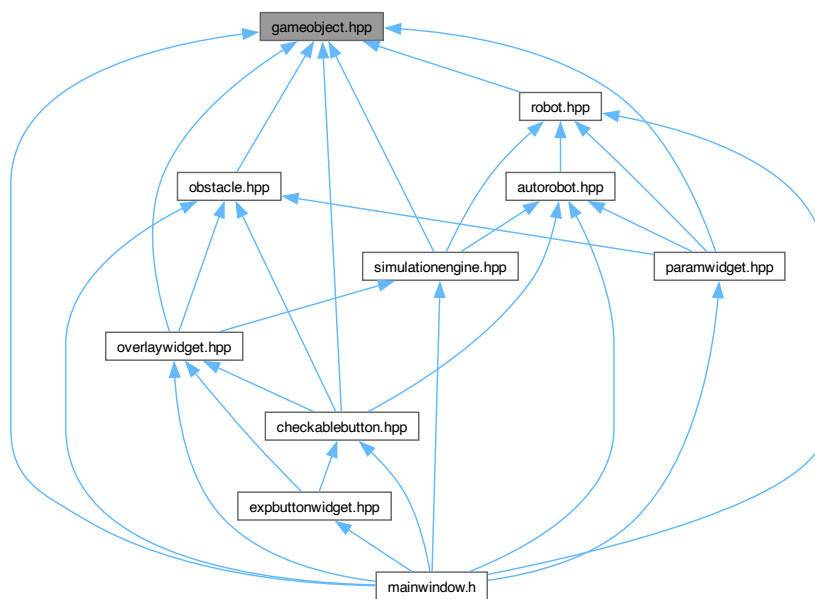
7.9 gameobject.hpp File Reference

This file contains the declaration of the [GameObject](#) class.

```
#include <QGraphicsItem>
#include <QJsonObject>
Include dependency graph for gameobject.hpp:
```



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



Classes

- class [GameObject](#)
A class to represent a game object in the simulation.

7.9.1 Detailed Description

This file contains the declaration of the [GameObject](#) class.

It is an abstract class that represents a game object in the simulation.

Authors

Tomáš Hobza, Jakub Všetečka

Date

02.05.2024

Definition in file [gameobject.hpp](#).

7.10 gameobject.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file gameobject.hpp
00003  * @brief This file contains the declaration of the GameObject class.
00004  * @details It is an abstract class that represents a game object in the simulation.
00005  * @authors Tomáš Hobza, Jakub Všetečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef GAMEOBJECT_HPP
00010 #define GAMEOBJECT_HPP
00011
00012 #include <QGraphicsItem>
00013 #include <QJsonObject>
00014
00015 /**
00016  * @class GameObject
00017  * @brief A class to represent a game object in the simulation.
00018  * @details This class provides an interface for creating and managing game objects.
00019  */
00020 class GameObject {
00021
00022 public:
00023     GameObject() = default;
00024     ~GameObject() = default;
00025
00026     /**
00027      * @brief Set the position of the game object.
00028      * @param x
00029      * @param y
00030      * @return void
00031      */
00032     virtual void setPos(qreal x, qreal y) = 0;
00033
00034     /**
00035      * @brief Paint the game object.
00036      * @param painter
00037      * @param option
00038      * @param widget
00039      * @return void
00040      */
00041     virtual void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) =
00042     0;
00043
00044     /**
00045      * @brief Get the position of the game object.
00046      * @return QPointF
00047      */
00047     virtual QPointF getPos() = 0;
00048
00049     /**
00050      * @brief Convert the game object to a JSON object.
00051      * @return QJsonObject
00052      */
00053     virtual QJsonObject toJSON() = 0;
00054
00055     /**
00056      * @brief Set the rotation of the game object.
00057      * @param angle
00058      * @return void
00059      */
00060     virtual void setRotation(qreal angle) = 0;
00061
00062     /**

```

```

00063     * @brief Get the rotation of the game object.
00064     * @return qreal
00065     */
00066     virtual qreal rotation() = 0;
00067
00068     /**
00069     * @brief Get the center of the game object.
00070     * @return QPointF
00071     */
00072     virtual QPointF getCenter() = 0;
00073 };
00074
00075 #endif // GAMEOBJECT_HPP

```

7.11 mainwindow.h File Reference

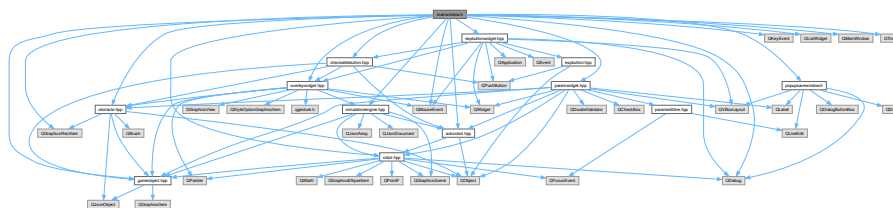
This file contains the declaration of the [MainWindow](#) class.

```

#include "autorobot.hpp"
#include "checkablebutton.hpp"
#include "expbuttonwidget.hpp"
#include "gameobject.hpp"
#include "obstacle.hpp"
#include "overlaywidget.hpp"
#include "paramwidget.hpp"
#include "popsavewindow.h"
#include "robot.hpp"
#include "simulationengine.hpp"
#include <QDebug>
#include <QGraphicsRectItem>
#include <QKeyEvent>
#include <QListWidget>
#include <QMainWindow>
#include <QMouseEvent>
#include <QTimer>

```

Include dependency graph for mainwindow.h:



Classes

- class [MainWindow](#)
A class to represent the main window of the application.

Namespaces

- namespace [Ui](#)

7.11.1 Detailed Description

This file contains the declaration of the [MainWindow](#) class.

It is a subclass of the [QMainWindow](#) class and represents the main window of the application.

Authors

Tomáš Hobza, Jakub Vřetečka

Date

02.05.2024

Definition in file [mainwindow.h](#).

7.12 mainwindow.h

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file mainwindow.h
00003  * @brief This file contains the declaration of the MainWindow class.
00004  * @details It is a subclass of the QMainWindow class and represents the main window of the
00005  * application.
00006  * @authors Tomáš Hobza, Jakub Vřetečka
00007  * @date 02.05.2024
00008  */
00009 #ifndef MAINWINDOW_H
00010 #define MAINWINDOW_H
00011
00012 #include "autorobot.hpp"
00013 #include "checkablebutton.hpp"
00014 #include "expbuttonwidget.hpp"
00015 #include "gameobject.hpp"
00016 #include "obstacle.hpp"
00017 #include "overlaywidget.hpp"
00018 #include "paramwidget.hpp"
00019 #include "popsavewindow.h"
00020 #include "robot.hpp"
00021 #include "simulationengine.hpp"
00022 #include <QDebug>
00023 #include <QGraphicsRectItem>
00024
00025 #include <QKeyEvent>
00026 #include <QListWidget>
00027 #include <QMainWindow>
00028 #include <QMouseEvent>
00029 #include <QTimer>
00030
00031 QT_BEGIN_NAMESPACE
00032 namespace Ui {
00033     class MainWindow;
00034 }
00035 QT_END_NAMESPACE
00036
00037 /**
00038  * @class MainWindow
00039  * @brief A class to represent the main window of the application.
00040  * @details This class inherits from QMainWindow and provides the main window of the application.
00041  * @see QMainWindow
00042  */
00043 class MainWindow : public QMainWindow {
00044     Q_OBJECT
00045
00046     public:
00047         MainWindow(QWidget *parent = nullptr);
00048         ~MainWindow();
00049         void setupAnimation();
00050
00051     private:

```

```

00052     /** @brief The UI object.*/
00053     Ui::MainWindow *ui;
00054
00055     /** @brief The simulation engine.*/
00056     SimulationEngine *simulationEngine;
00057
00058     /** @brief The list widget.*/
00059     QListWidget *listWidget;
00060
00061     void initScene();
00062     void updateAnimation(); // Method to update the animation
00063
00064 protected:
00065     /** @brief The param widget.*/
00066     ParamWidget *paramWidget;
00067
00068     /** @brief The expandable button widget.*/
00069     ExpandableButtonWidget *expandableWidget;
00070
00071     /** @brief The overlay widget.*/
00072     OverlayWidget *overlay;
00073
00074 private slots:
00075     /**
00076      * @brief Slot to handle the save button click event.
00077      *
00078      * @return void
00079      */
00080     void saveSimulation();
00081
00082     /**
00083      * @brief Overridden event filter method to handle key press events.
00084      *
00085      * @param object The object that the event is being filtered for
00086      * @param event The event that is being filtered
00087      * @return bool Whether the event was handled
00088      */
00089     bool eventFilter(QObject *object, QEvent *event) override;
00090
00091     /**
00092      * @brief Overridden show event method to handle the show event.
00093      *
00094      * @param event The show event
00095      * @return void
00096      */
00097     void showEvent(QShowEvent *event) override;
00098
00099     /**
00100      * @brief Overridden resize event method to handle the resize event.
00101      *
00102      * @param event The resize event
00103      * @return void
00104      */
00105     void resizeEvent(QResizeEvent *event) override;
00106
00107     /**
00108      * @brief Overridden close event method to handle the close event.
00109      *
00110      * @param event The close event
00111      * @return void
00112      */
00113     void mouseDoubleClickEvent(QMouseEvent *event) override;
00114
00115     /**
00116      * @brief Overridden mouse press event method to handle the mouse press event.
00117      *
00118      * @param event The mouse press event
00119      * @return void
00120      */
00121     void mousePressEvent(QMouseEvent *event) override;
00122
00123     /**
00124      * @brief Overridden mouse move event method to handle the mouse move event.
00125      *
00126      * @param event The mouse move event
00127      * @return void
00128      */
00129     void mouseMoveEvent(QMouseEvent *event) override;
00130
00131     /**
00132      * @brief Overridden mouse release event method to handle the mouse release event.
00133      *
00134      * @param event The mouse release event
00135      * @return void
00136      */
00137     void mouseReleaseEvent(QMouseEvent *event) override;
00138

```



```

00139     /**
00140      * @brief Slot to handle the horizontal slider value changed event.
00141      *
00142      * @param value The new value of the slider
00143      * @return void
00144      */
00145     void on_horizontalSlider_valueChanged(int value);
00146
00147     /**
00148      * @brief Slot to handle toggling the list.
00149      *
00150      * @return void
00151      */
00152     void toggleList();
00153
00154     /**
00155      * @brief Slot to handle the item double click event from the list.
00156      *
00157      * @param item The item that was double clicked
00158      * @return void
00159      */
00160     void handleItemDoubleClick(QListWidgetItem *item);
00161
00162     /**
00163      * @brief Slot to handle clear button click event.
00164      *
00165      * @return void
00166      */
00167     void on_pushButton_clicked();
00168
00169     /**
00170      * @brief Slot to handle rotate anticlockwise button click event.
00171      *
00172      * @return void
00173      */
00174     void goLeft();
00175
00176     /**
00177      * @brief Slot to handle stop rotating button click event.
00178      *
00179      * @return void
00180      */
00181     void stopRotating();
00182
00183     /**
00184      * @brief Slot to handle rotate clockwise button click event.
00185      *
00186      * @return void
00187      */
00188     void goRight();
00189
00190     /**
00191      * @brief Slot to handle move forward button click event.
00192      *
00193      * @return void
00194      */
00195     void goStraight();
00196
00197     /**
00198      * @brief Slot to handle stop moving button click event.
00199      *
00200      * @return void
00201      */
00202     void stopMoving();
00203 };
00204 #endif // MAINWINDOW_H

```

7.13 obstacle.hpp File Reference

This file contains the declaration of the [Obstacle](#) class.

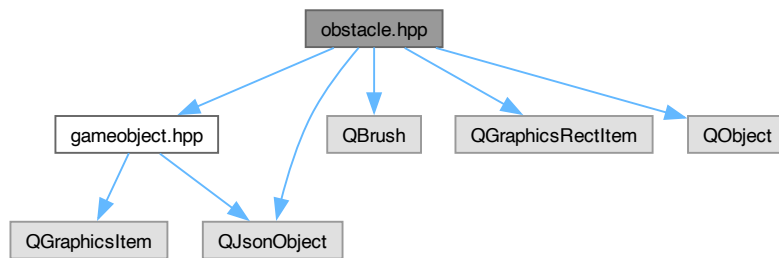
```

#include "gameobject.hpp"
#include <QBrush>
#include <QGraphicsRectItem>
#include <QJsonObject>

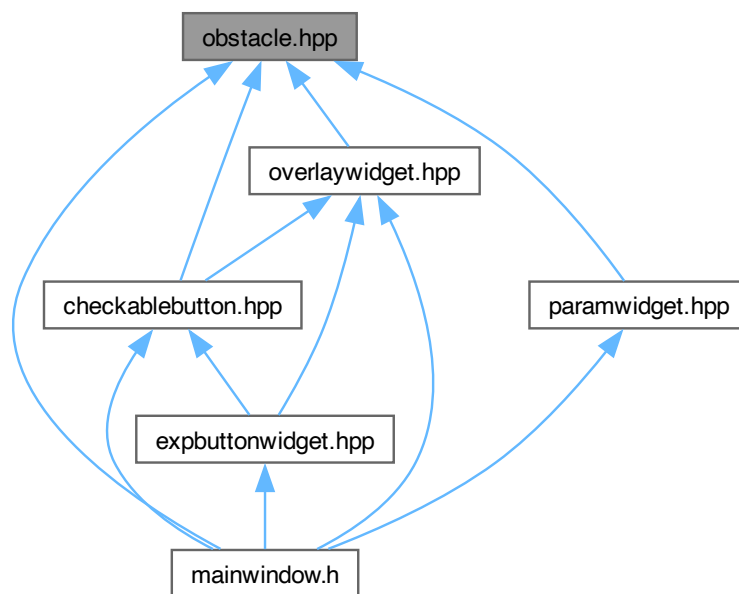
```

```
#include <QObject>
```

Include dependency graph for obstacle.hpp:



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



Classes

- class [Obstacle](#)
A class to represent an obstacle.

7.13.1 Detailed Description

This file contains the declaration of the [Obstacle](#) class.

It is a subclass of the [QGraphicsRectItem](#) class and represents an obstacle in the game.

Authors

Tomáš Hobza, Jakub Všetečka

Date

02.05.2024

Definition in file [obstacle.hpp](#).

7.14 obstacle.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file obstacle.hpp
00003  * @brief This file contains the declaration of the Obstacle class.
00004  * @details It is a subclass of the QGraphicsRectItem class and represents an obstacle in the game.
00005  * @authors Tomáš Hobza, Jakub Všetečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef OBSTACLE_HPP
00010 #define OBSTACLE_HPP
00011
00012 #include "gameobject.hpp"
00013 #include <QBrush>
00014 #include <QGraphicsRectItem>
00015 #include <QJsonObject>
00016 #include <QObject>
00017
00018 /**
00019  * @class Obstacle
00020  * @brief A class to represent an obstacle.
00021  * @details This class inherits from QGraphicsRectItem and GameObject. It represents an obstacle in a
00022  * game.
00023  */
00024 class Obstacle : public QObject, public QGraphicsRectItem, public GameObject {
00025     Q_OBJECT
00026 public:
00027     /**
00028      * @brief Default constructor.
00029      * @param parent The parent QGraphicsItem.
00030      * @return void
00031      */
00032     Obstacle(QGraphicsItem *parent = nullptr);
00033
00034     /**
00035      * @brief Copy constructor.
00036      * @param Obstacle The Obstacle object to copy.
00037      * @return void
00038      */
00039     Obstacle(const Obstacle &)
00040         : QGraphicsRectItem() {}
00041
00042     /**
00043      * @brief Destructor.
00044      */
00045     ~Obstacle();
00046
00047     /**
00048      * @brief Set the position of the obstacle.
00049      * @param x The x-coordinate of the position.
00050      * @param y The y-coordinate of the position.
00051      * @return void
00052      */
00053     void setPos(qreal x, qreal y) override;
00054
00055     /**
00056      * @brief Set the rotation of the obstacle.
00057      * @param angle The angle of the rotation.
00058      * @return void
00059      */
00060     void setRotation(qreal angle) override;
00061
00062     /**

```

```

00063     * @brief Get the rotation of the obstacle.
00064     * @return The rotation of the obstacle as a qreal.
00065     */
00066     qreal rotation() override { return QGraphicsRectItem::rotation(); }
00067
00068     /**
00069     * @brief Get the center of the obstacle.
00070     * @return The center of the obstacle as a QPointF.
00071     */
00072     QPointF getCenter() override { return boundingRect().center(); }
00073
00074     /**
00075     * @brief Paint the obstacle.
00076     * @param painter Pointer to the QPainter object.
00077     * @param option Pointer to the QStyleOptionGraphicsItem object.
00078     * @param widget Pointer to the QWidget object.
00079     */
00080     void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
00081
00082     /**
00083     * @brief Get the position of the obstacle.
00084     * @return The position of the obstacle as a QPointF object.
00085     */
00086     QPointF getPos() override;
00087
00088     /**
00089     * @brief Convert the obstacle to a JSON object.
00090     * @return The obstacle as a QJsonObject.
00091     */
00092     QJsonObject toJSON() override;
00093
00094     /**
00095     * @brief Create an Obstacle object from a JSON object.
00096     * @param json The QJsonObject to convert.
00097     * @return A pointer to the created Obstacle object.
00098     */
00099     static Obstacle *fromJSON(const QJsonObject &json);
00100
00101     signals:
00102     /**
00103     * @brief Signal emitted when the parameters of the obstacle are updated.
00104     * @return void
00105     */
00106     void paramsUpdated();
00107
00108     /**
00109     * @brief Signal emitted when the obstacle is removed.
00110     * @return void
00111     */
00112     void obstacleSepuku();
00113 };
00114
00115 #endif // OBSTACLE_HPP

```

7.15 overlaywidget.hpp File Reference

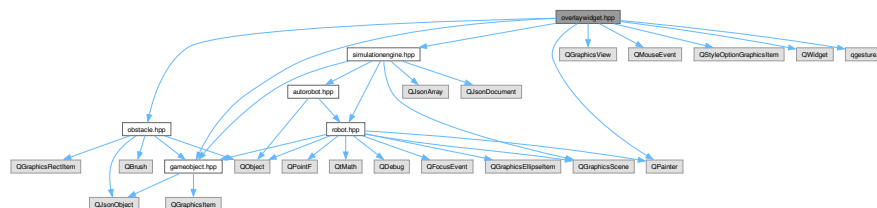
This file contains the declaration of the [OverlayWidget](#) class.

```

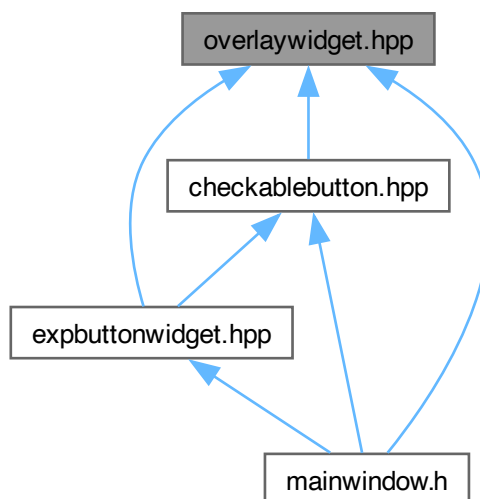
#include "gameobject.hpp"
#include "obstacle.hpp"
#include "simulationengine.hpp"
#include <QGraphicsView>
#include <QMouseEvent>
#include <QPainter>
#include <QStyleOptionGraphicsItem>
#include <QWidget>
#include <qgesture.h>

```

Include dependency graph for overlaywidget.hpp:



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



Classes

- class **OverlayWidget**

A class to represent an overlay widget.

7.15.1 Detailed Description

This file contains the declaration of the [OverlayWidget](#) class.

It is a subclass of the [QWidget](#) class and represents an overlay widget.

Authors

Tomáš Hobza, Jakub Vřetečka

Definition in file [overlaywidget.hpp](#).

7.16 overlaywidget.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file overlaywidget.hpp
00003  * @brief This file contains the declaration of the OverlayWidget class.
00004  * @details It is a subclass of the QWidget class and represents an overlay widget.
00005  * @authors Tomáš Hobza, Jakub Vřetečka
00006  */
00007
00008 #ifndef OVERLAYWIDGET_HPP
00009 #define OVERLAYWIDGET_HPP
00010
00011 #include "gameobject.hpp"
00012 #include "obstacle.hpp"
00013 #include "simulationengine.hpp"
00014 #include <QGraphicsView>
00015 #include <QMouseEvent>
00016 #include <QPainter>
00017 #include <QStyleOptionGraphicsItem>
00018 #include <QWidget>
00019 #include <qgesture.h>
00020
00021 /**
00022  * @class OverlayWidget
00023  * @brief A class to represent an overlay widget.
00024  * @details This class provides an interface for creating and managing overlay widgets.
00025  * @see QWidget
00026  */
00027 class OverlayWidget : public QWidget {
00028 public:
00029     /**
00030      * @brief Construct a new Overlay Widget object.
00031      *
00032      * @param parent The parent widget. Default is nullptr.
00033      * @param simEng The simulation engine. Default is nullptr.
00034      * @param graphView The graphics view. Default is nullptr.
00035      */
00036     explicit OverlayWidget(QWidget *parent = nullptr, SimulationEngine *simEng = nullptr,
00037                             QGraphicsView *graphView = nullptr);
00037
00038     /**
00039      * @brief Try grab the object based on the mouse position.
00040      * @param event The mouse event.
00041      * @return void
00042      */
00043     void trySetSail(QMouseEvent *event);
00044
00045     /**
00046      * @brief Drag the object based on the mouse position in the overlay.
00047      * @param event The mouse event.
00048      * @return void
00049      */
00050     void navigateTheSea(QMouseEvent *event);
00051
00052     /**
00053      * @brief Anchor the object based on the mouse position back to scene.
00054      * @return void
00055      */
00056     void anchor();
00057
00058     /**
00059      * @brief Get the time constant of the simulation engine.
00060      * @return qreal* The time constant of the simulation engine.
00061      */
00062     qreal *getTimeConstant() { return simEng->getTimeConstant(); }
00063
00064     /**
00065      * @brief Get the active object.
00066      * @return GameObject* The active object.
00067      */
00068     void setActiveObject(GameObject *obj) { activeObject = obj; }
00069
00070     /**
00071      * @brief Get the last mouse position.
00072      * @return QPoint The last mouse position.
00073      */
00074     void setLastMousePos(QPoint pos) { lastMousePos = pos; }
00075
00076 protected:
00077     /** @brief The active object. */
00078     GameObject *activeObject;
00079
00080     /** @brief The last mouse position. */
00081     QPoint lastMousePos;

```

```

00082
00083     QPoint offset;
00084
00085     /** @brief The simulation engine. */
00086     SimulationEngine *simEng;
00087
00088     /** @brief The graphics view. */
00089     QGraphicsView *graphView;
00090
00091     /** @brief The option for the graphics item. */
00092     QStyleOptionGraphicsItem option;
00093
00094     /**
00095      * @brief Override the mousePressEvent method.
00096      * @param event The mouse event.
00097      * @return void
00098      */
00099     void paintEvent(QPaintEvent *event) override;
00100
00101     /**
00102      * @brief Convert the point to the rotated system.
00103      * @param point The point in the scene.
00104      * @param angle The angle of the rotation.
00105      * @return QPoint The point in the rotated system.
00106      */
00107     QPoint convertToRotatedSystem(QPoint point, qreal angle);
00108
00109     /**
00110      * @brief Convert the point from the rotated system.
00111      * @param point The point in the rotated system.
00112      * @param angle The angle of the rotation.
00113      * @return QPoint The point in the rotated system.
00114      */
00115     QPoint convertFromRotatedSystem(QPoint point, qreal angle);
00116 };
00117
00118 #endif // OVERLAYWIDGET_HPP

```

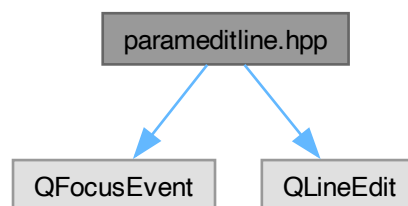
7.17 parameditline.hpp File Reference

This file contains the declaration of the [ParamEditLine](#) class.

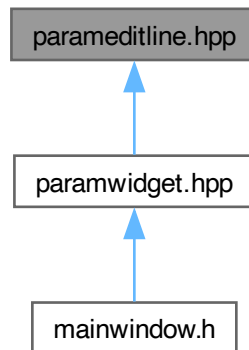
```
#include <QFocusEvent>
```

```
#include <QLineEdit>
```

Include dependency graph for parameditline.hpp:



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



Classes

- class [ParamEditLine](#)
A class to represent a line edit widget for editing parameters.

7.17.1 Detailed Description

This file contains the declaration of the [ParamEditLine](#) class.

It is a subclass of the [QLineEdit](#) class and represents a line edit widget for editing parameters.

Authors

Tomáš Hobza, Jakub Všeckčka

Date

03.05.2024

Definition in file [parameditline.hpp](#).

7.18 parameditline.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file parameditline.hpp
00003  * @brief This file contains the declaration of the ParamEditLine class.
00004  * @details It is a subclass of the QLineEdit class and represents a line edit widget for editing
00005  * parameters.
00006  * @authors Tomáš Hobza, Jakub Vřetečka
00007  * @date 03.05.2024
00008  */
00009 #ifndef PARAMEDITLINE_HPP
00010 #define PARAMEDITLINE_HPP
00011
00012 #include <QFocusEvent>
00013 #include <QLineEdit>
00014
00015 /**
00016  * @class ParamEditLine
00017  * @brief A class to represent a line edit widget for editing parameters.
00018  * @details This class inherits from QLineEdit and provides a line edit widget for editing parameters.
00019  * @see QLineEdit
00020  */
00021 class ParamEditLine : public QLineEdit {
00022     Q_OBJECT
00023
00024 public:
00025     /**
00026      * @brief Default constructor.
00027      * @param parent The parent widget.
00028      */
00029     explicit ParamEditLine(QWidget *parent = nullptr)
00030         : QLineEdit(parent) {}
00031
00032     /**
00033      * @brief Overridden focusInEvent method.
00034      * @param event The focus event.
00035      * @return void
00036      */
00037     void focusInEvent(QFocusEvent *event) override {
00038         QLineEdit::focusOutEvent(event);
00039         emit focusIn();
00040     }
00041
00042     /**
00043      * @brief Overridden focusOutEvent method.
00044      * @param event The focus event.
00045      * @return void
00046      */
00047     void focusOutEvent(QFocusEvent *event) override {
00048         QLineEdit::focusOutEvent(event);
00049         emit focusOut();
00050     }
00051
00052 signals:
00053     /**
00054      * @brief Signal emitted when the line edit widget gains focus.
00055      * @return void
00056      */
00057     void focusIn();
00058
00059     /**
00060      * @brief Signal emitted when the line edit widget loses focus.
00061      * @return void
00062      */
00063     void focusOut();
00064 };
00065
00066 #endif // PARAMEDITLINE_HPP

```

7.19 paramwidget.hpp File Reference

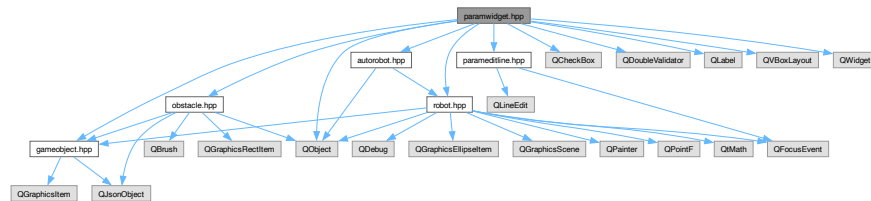
This file contains the declaration of the [ParamWidget](#) class.

```

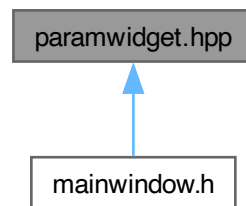
#include "autorobot.hpp"
#include "gameobject.hpp"

```

```
#include "obstacle.hpp"
#include "parameditline.hpp"
#include "robot.hpp"
#include <QCheckBox>
#include <QDoubleValidator>
#include <QLabel>
#include <QObject>
#include <QVBoxLayout>
#include <QWidget>
Include dependency graph for paramwidget.hpp:
```



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



Classes

- class [ParamWidget](#)

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

7.19.1 Detailed Description

This file contains the declaration of the [ParamWidget](#) class.

It is a subclass of the [QWidget](#) class and represents a widget for editing parameters of game objects.

Authors

Tomáš Hobza, Jakub Všecká

Date

03.05.2024

Definition in file [paramwidget.hpp](#).

7.20 paramwidget.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file paramwidget.hpp
00003  * @brief This file contains the declaration of the ParamWidget class.
00004  * @details It is a subclass of the QWidget class and represents a widget for editing parameters of
game objects.
00005  * @authors Tomáš Hobza, Jakub Všeťka
00006  * @date 03.05.2024
00007  */
00008
00009 #ifndef PARAMWIDGET_HPP
00010 #define PARAMWIDGET_HPP
00011
00012 #include "autorobot.hpp"
00013 #include "gameobject.hpp"
00014 #include "obstacle.hpp"
00015 #include "parameditline.hpp"
00016 #include "robot.hpp"
00017 #include <QCheckBox>
00018 #include <QDoubleValidator>
00019 #include <QLabel>
00020 #include <QObject>
00021 #include <QVBoxLayout>
00022 #include <QWidget>
00023
00024 /**
00025  * @class ParamWidget
00026  * @brief A class to represent a widget for editing parameters of game objects.
00027  * @details This class inherits from QWidget and provides a widget for editing parameters of game
objects.
00028  * @see QWidget
00029  */
00030 class ParamWidget : public QWidget {
00031     Q_OBJECT
00032
00033 public:
00034     /**
00035      * @brief Default constructor.
00036      * @param parent The parent widget.
00037      */
00038     explicit ParamWidget(QWidget *parent = nullptr);
00039
00040     /**
00041      * @brief Set the game object whose parameters will be displayed.
00042      * @param object The game object.
00043      * @return void
00044      */
00045     void stalk(AutoRobot *robot);
00046
00047     /**
00048      * @brief Set the game object whose parameters will be displayed.
00049      * @param object The game object.
00050      * @return void
00051      */
00052     void stalk(Obstacle *obstacle);
00053
00054     /**
00055      * @brief Set the game object whose parameters will be displayed.
00056      * @param object The game object.
00057      * @return void
00058      */
00059     void stalk(Robot *robot);
00060
00061     /**
00062      * @brief Stop editing the parameters of the game object.
00063      * @return void
00064      */
00065     void stopStalking();
00066
00067 private:
00068     /** @brief The layout of the widget. */
00069     QVBoxLayout *layout;
00070
00071     /** @brief The game object whose parameters are being displayed. */
00072     GameObject *stalkedObject = nullptr;
00073
00074     /** @brief Whether the widget should keep updating the parameters of the game object. */
00075     bool keepUpdating = true;
00076
00077     /** @brief The validator for the number input. */
00078     QDoubleValidator *numberValidator;
00079
00080     /** @brief The labels and line edit widgets for editing the parameters. */

```

```

00081     QLabel *labelDetectionDistance;
00082     ParamEditLine *detectionDistance;
00083     QLabel *labelAngleToRotate;
00084     ParamEditLine *angleToRotate;
00085     QLabel *labelDirection;
00086     QCheckBox *direction;
00087     QLabel *labelSpeed;
00088     ParamEditLine *speed;
00089     QLabel *labelRadius;
00090     ParamEditLine *radius;
00091     QLabel *labelAngle;
00092     ParamEditLine *angle;
00093     QLabel *labelSize;
00094     ParamEditLine *size;
00095
00096     /**
00097      * @brief Set up the line edit widget for editing a parameter.
00098      * @param lineEdit The line edit widget.
00099      * @param label The label for the line edit widget.
00100      * @return void
00101      */
00102     void setUpEditLine(ParamEditLine *lineEdit, QLabel *label);
00103
00104     /**
00105      * @brief Show the parameters of the game object.
00106      * @param robot The robot whose parameters will be displayed.
00107      * @return void
00108      */
00109     void show(Robot *robot);
00110
00111     /**
00112      * @brief Show the parameters of the game object.
00113      * @param robot The robot whose parameters will be displayed.
00114      * @return void
00115      */
00116     void show(AutoRobot *robot);
00117
00118     /**
00119      * @brief Show the parameters of the game object.
00120      * @param obstacle The obstacle whose parameters will be displayed.
00121      * @return void
00122      */
00123     void show(Obstacle *obstacle);
00124
00125     /**
00126      * @brief Hide the widget.
00127      * @return void
00128      */
00129     void hide();
00130
00131     /**
00132      * @brief Disconnect the widget from the game object.
00133      * @return void
00134      */
00135     void disconnectStalkedObject();
00136
00137 private slots:
00138     /**
00139      * @brief Signal to set the detection distance of the game object.
00140      * @return void
00141      */
00142     void setDetectionDistance();
00143
00144     /**
00145      * @brief Signal to set the angle to rotate of the game object.
00146      * @return void
00147      */
00148     void setAngleToRotate();
00149
00150     /**
00151      * @brief Signal to set the direction of the game object.
00152      * @return void
00153      */
00154     void setDirection();
00155
00156     /**
00157      * @brief Signal to set the speed of the game object.
00158      * @return void
00159      */
00160     void setSpeed();
00161
00162     /**
00163      * @brief Signal to set the radius of the game object.
00164      * @return void
00165      */
00166     void setRadius();
00167

```

```

00168     /**
00169     * @brief Signal to set the angle of the game object.
00170     * @return void
00171     */
00172     void setAngle();
00173
00174     /**
00175     * @brief Signal to set the size of the game object.
00176     * @return void
00177     */
00178     void setSize();
00179
00180     /**
00181     * @brief Signal to update the parameters of the game object.
00182     * @return void
00183     */
00184     inline void focusIn() { keepUpdating = false; }
00185
00186     /**
00187     * @brief Signal to update the parameters of the game object.
00188     * @return void
00189     */
00190     inline void focusOut() { keepUpdating = true; }
00191
00192     /**
00193     * @brief Update the parameters of the game object.
00194     * @return void
00195     */
00196     void updateAutoRobot();
00197
00198     /**
00199     * @brief Update the parameters of the game object.
00200     * @return void
00201     */
00202     void updateObstacle();
00203
00204     /**
00205     * @brief Update the parameters of the game object.
00206     * @return void
00207     */
00208     void updateRobot();
00209 };
00210
00211 #endif // PARAMWIDGET_HPP

```

7.21 popupsavewindow.h File Reference

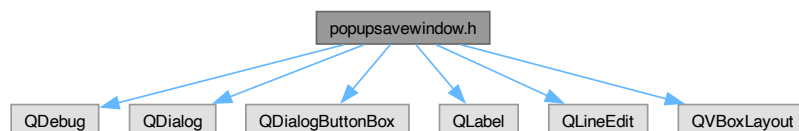
This file contains the declaration of the [PopupSaveWindow](#) class.

```

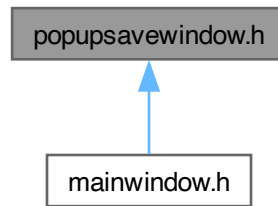
#include <QDebug>
#include <QDialog>
#include <QDialogButtonBox>
#include <QLabel>
#include <QLineEdit>
#include <QVBoxLayout>

```

Include dependency graph for popupsavewindow.h:



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



Classes

- class [PopupSaveWindow](#)
A class to represent a popup save window.

7.21.1 Detailed Description

This file contains the declaration of the [PopupSaveWindow](#) class.

It is a subclass of the [QDialog](#) class and represents a popup save window.

Authors

Tomáš Hobza, Jakub Všeťečka

Date

02.05.2024

Definition in file [popupsavewindow.h](#).

7.22 popupsavewindow.h

[Go to the documentation of this file.](#)

```
00001 /**
00002  * @file popupsavewindow.h
00003  * @brief This file contains the declaration of the PopupSaveWindow class.
00004  * @details It is a subclass of the QDialog class and represents a popup save window.
00005  * @authors Tomáš Hobza, Jakub Všeťečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef POPUPSAVEWINDOW_H
00010 #define POPUPSAVEWINDOW_H
00011
00012 #include <QDebug>
00013 #include <QDialog>
00014 #include <QDialogButtonBox>
00015 #include <QLabel>
```

```

00016 #include <QLineEdit>
00017 #include <QVBoxLayout>
00018
00019 /**
00020  * @class PopupSaveWindow
00021  * @brief A class to represent a popup save window.
00022  * @details This class provides an interface for creating and managing a popup save window.
00023  * @see QDialog
00024  */
00025 class PopupSaveWindow : public QDialog {
00026     Q_OBJECT
00027
00028 public:
00029     /**
00030      * @brief Construct a new Popup Save Window object.
00031      * @param parent The parent widget. Default is nullptr.
00032      */
00033     explicit PopupSaveWindow(QWidget *parent = nullptr);
00034     ~PopupSaveWindow();
00035
00036     /**
00037      * @brief Get the entered text.
00038      * @return QString The entered text.
00039      */
00040     QString getEnteredText() { return enteredText; }
00041
00042 private:
00043     /** @brief The entered text. */
00044     QString enteredText;
00045
00046     /** @brief The line edit widget. */
00047     QLineEdit *lineEdit;
00048
00049 private slots:
00050
00051     /**
00052      * @brief Slot to handle the ok button click event.
00053      * @return void
00054      */
00055     void onOkClicked();
00056 };
00057
00058 #endif // POPUPSAVEWINDOW_H

```

7.23 robot.hpp File Reference

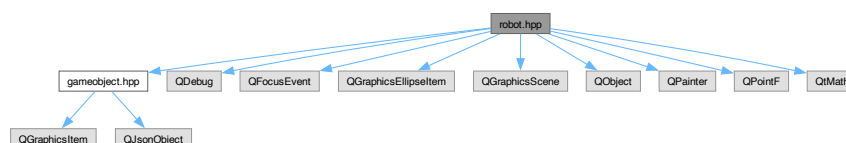
This file contains the declaration of the [Robot](#) class.

```

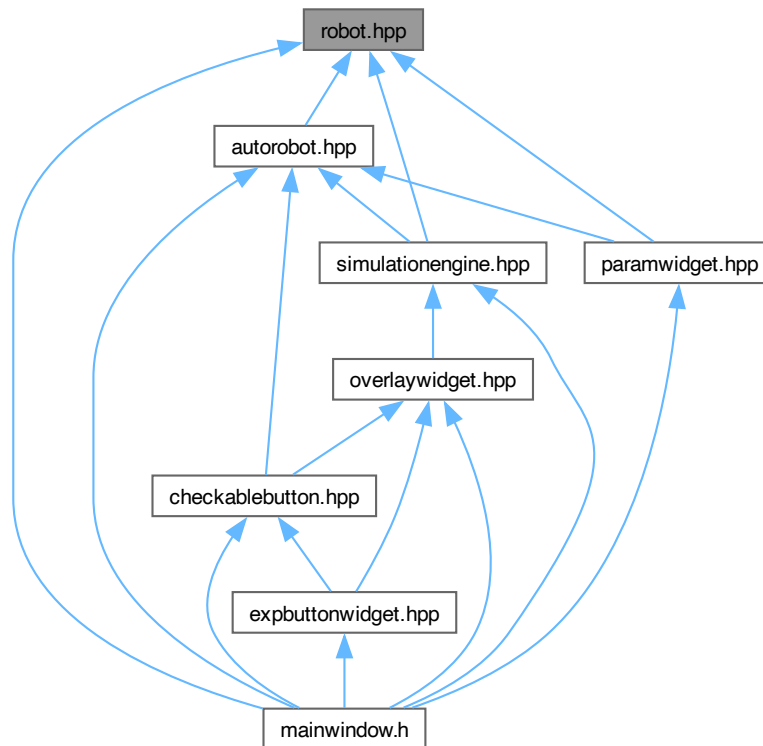
#include "gameobject.hpp"
#include <QDebug>
#include <QFocusEvent>
#include <QGraphicsEllipseItem>
#include <QGraphicsScene>
#include <QObject>
#include <QPainter>
#include <QPointF>
#include <QtMath>

```

Include dependency graph for robot.hpp:



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



Classes

- class [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.

Macros

- `#define` [BODY_COLLISION_MARGIN](#) 1

7.23.1 Detailed Description

This file contains the declaration of the [Robot](#) class.

It is a subclass of the [QGraphicsEllipseItem](#) class and represents a robot in the simulation.

Authors

Tomáš Hobza, Jakub Vřetečka

Date

02.05.2024

Definition in file [robot.hpp](#).

7.23.2 Macro Definition Documentation

7.23.2.1 BODY_COLLISION_MARGIN

```
#define BODY_COLLISION_MARGIN 1
```

Definition at line 22 of file [robot.hpp](#).

7.24 robot.hpp

[Go to the documentation of this file.](#)

```
00001 /**
00002  * @file robot.hpp
00003  * @brief This file contains the declaration of the Robot class.
00004  * @details It is a subclass of the QGraphicsEllipseItem class and represents a robot in the
simulation.
00005  * @authors Tomáš Hobza, Jakub Všetečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef ROBOT_HPP
00010 #define ROBOT_HPP
00011
00012 #include "gameobject.hpp"
00013 #include <QDebug>
00014 #include <QFocusEvent>
00015 #include <QGraphicsEllipseItem>
00016 #include <QGraphicsScene>
00017 #include <QObject>
00018 #include <QPainter>
00019 #include <QPointF>
00020 #include <QtMath>
00021
00022 #define BODY_COLLISION_MARGIN 1
00023
00024 /**
00025  * @brief A class to represent a robot in the simulation. By default, the robot is a circle with a
line drawn to represent its direction.
00026  */
00027 class Robot : public QObject, public QGraphicsEllipseItem, public GameObject {
00028     Q_OBJECT
00029
00030     public:
00031         /**
00032          * @brief Enum to represent the direction of rotation of the robot.
00033          */
00034         enum RotationDirection {
00035             Left = -1, // Counter-clockwise
00036             None = 0, // No rotation
00037             Right = 1 // Clockwise
00038         };
00039
00040         enum { Type = QGraphicsItem::UserType + 1 };
00041
00042         /**
00043          * @brief Default constructor.
00044          * @param parent The parent QGraphicsItem.
00045          * @param timeConstant The time constant of the simulation.
00046          * @return void
00047          * @details The time constant is used to calculate the speed of the robot.
00048          */
00049         Robot(QGraphicsItem *parent = nullptr, qreal *timeConstant = nullptr);
00050
00051         ~Robot();
00052
00053         /** Override the paint method to draw a line showing the direction of the robot */
00054         virtual void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
override;
00055
00056         /** Override setPos to adjust to center-based positioning */
00057         void setPos(const QPointF &pos);
00058
00059         /** Overload setPos to accept x and y coordinates */
00060         void setPos(qreal x, qreal y) override;
00061
00062         /** Override the boundingRect method to adjust the bounding rectangle */
00063         virtual QRectF boundingRect() const override;
```

```

00064
00065     /** Override pos to adjust to center-based positioning */
00066     QPointF pos();
00067
00068     qreal getRadius() const;
00069
00070     /**
00071      * @brief Set the move speed of the robot.
00072      *
00073      * @param speed
00074      */
00075     void setMoveSpeed(qreal speed);
00076
00077     /**
00078      * @brief Get the move speed of the robot.
00079      *
00080      * @return qreal
00081      */
00082     qreal getMoveSpeed();
00083
00084     /**
00085      * @brief Set the rotation speed of the robot.
00086      *
00087      * @param speed
00088      */
00089     void setRotationSpeed(qreal speed);
00090
00091     /**
00092      * @brief Get the rotation speed of the robot.
00093      *
00094      * @return qreal
00095      */
00096     qreal getRotationSpeed();
00097
00098     /**
00099      * @brief Allow the robot to be moved by setting the isMoving flag to true.
00100      */
00101     void startMoving();
00102
00103     /**
00104      * @brief Stop the robot from moving by setting the isMoving flag to false.
00105      */
00106     void stopMoving();
00107
00108     /**
00109      * @brief Start rotating the robot in the given direction.
00110      *
00111      * @param direction
00112      */
00113     void startRotating(RotationDirection direction);
00114
00115     /**
00116      * @brief Stop the robot from rotating by setting the isRotating flag to None.
00117      */
00118     void stopRotating();
00119
00120     /**
00121      * @brief Get the direction vector of the robot.
00122      *
00123      * @return `QPointF` - Normalized vector representing the direction of the robot on the x and y
00124      axes
00125      */
00126     QPointF getDirectionVector();
00127
00128     /**
00129      * @brief Check if the robot will collide with any other item in the scene or the scene boundaries
00130      if it moves by the given vector.
00131      *
00132      * @param moveVector The vector by which the robot will move
00133      * @param allowAnticollision Flag to indicate if anticollision is allowed
00134      * @return `true` - if the robot will collide; `false` - if the robot will not collide
00135      */
00136     virtual bool willCollide(QPointF directionVector, qreal magnitude, bool allowAnticollision =
00137     false);
00138
00139     /**
00140      * @brief Move the robot based on its current direction and speed. Returns true if the robot
00141      moved, false if it didn't (e.g. if it hit a boundary).
00142      *
00143      * @return true
00144      * @return false
00145      */
00146     virtual bool move();
00147
00148     /**
00149      * @brief Get the type of the robot.
00150      *
00151      * @return int

```

```

00147     */
00148     int type() const override { return Type; }
00149
00150     /**
00151      * @brief Get the position of the robot.
00152      * @return QPointF
00153      */
00154     QPointF getPos() override;
00155
00156     /**
00157      * @brief Convert the robot to a JSON object.
00158      * @return QJsonObject
00159      */
00160     virtual QJsonObject toJSON() override;
00161
00162     /**
00163      * @brief Create a Robot object from a JSON object.
00164      * @param object The JSON object.
00165      * @param timeConstant The time constant of the simulation.
00166      * @return Robot*
00167      */
00168     static Robot *fromJSON(const QJsonObject &object, qreal *timeConstant);
00169
00170     /**
00171      * @brief Toggle the active state of the robot.
00172      * @details If the robot is active, it will be drawn with a light gray fill. If it is inactive, it
00173      * will be drawn with a transparent fill.
00174      * @return void
00175      */
00176     inline void toggleActive() {
00177         active = !active;
00178         active ? setBrush(QBrush(Qt::lightGray)) : setBrush(QBrush(Qt::transparent));
00179     }
00180
00181     /**
00182      * @brief Check if the robot is active.
00183      * @return bool
00184      */
00185     inline bool isActive() { return active; }
00186
00187     /**
00188      * @brief Get the angle of the robot.
00189      * @return qreal
00190      */
00191     qreal getAngle() { return rotation(); }
00192
00193     /**
00194      * @brief Set the angle of the robot.
00195      * @param angle The angle to set.
00196      * @return void
00197      */
00198     void setRadius(qreal radius);
00199
00200     /**
00201      * @brief Get the center of the robot.
00202      * @return QPointF
00203      */
00204     QPointF getCenter() override { return boundingRect().center(); }
00205
00206     /**
00207      * @brief Get the time constant of the simulation.
00208      * @return qreal
00209      */
00210     qreal rotation() override {
00211         return QGraphicsEllipseItem::rotation();
00212     }
00213
00214     /**
00215      * @brief Set the rotation of the robot.
00216      * @param angle The angle to set.
00217      * @return void
00218      */
00219     void setRotation(qreal angle) override {
00220         QGraphicsEllipseItem::setRotation(angle);
00221     }
00222
00223     signals:
00224
00225     /**
00226      * @brief Signal emitted when the parameters of the robot are updated.
00227      * @return void
00228      */
00229     void paramsUpdated();
00230
00231     /**
00232      * @brief Signal emitted when the robot is removed.
00233      * @return void

```

```

00233     */
00234     void robotSepuku();
00235
00236 protected:
00237     /** @brief The speed of the robot */
00238     qreal move_speed = 1;
00239     /** @brief The speed of the rotation of the robot */
00240     qreal rotation_speed = 1;
00241
00242     /** @brief Flag to indicate if the robot is moving */
00243     bool isMoving = false;
00244
00245     /** @brief Flag to indicate the direction of rotation */
00246     RotationDirection isRotating = RotationDirection::None;
00247
00248     /** @brief The time constant of the simulation */
00249     qreal *timeConstant = nullptr;
00250
00251     /** @brief The radius of the robot */
00252     void keyReleaseEvent(QKeyEvent *event);
00253
00254     /**
00255      * @brief Overridden keyPressEvent method.
00256      * @details This method is called when a key is pressed while the robot is focused.
00257      * @param event The key event.
00258      * @return void
00259      */
00260     void keyPressEvent(QKeyEvent *event);
00261
00262 private:
00263     /** @brief Flag to indicate if the robot is active */
00264     bool active = false;
00265 };
00266
00267 #endif // ROBOT_HPP

```

7.25 simulationengine.hpp File Reference

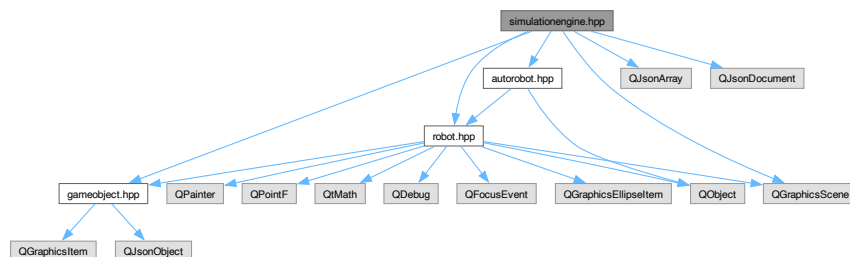
This file contains the declaration of the [SimulationEngine](#) class.

```

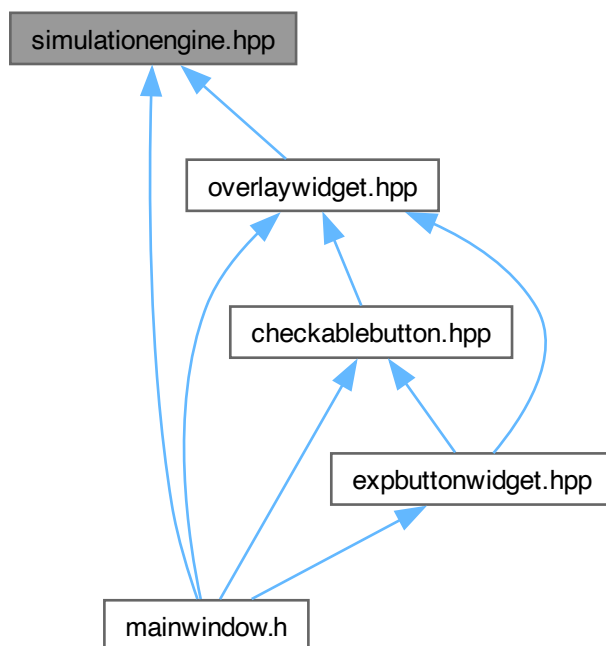
#include "autorobot.hpp"
#include "gameobject.hpp"
#include "robot.hpp"
#include <QGraphicsScene>
#include <QJsonArray>
#include <QJsonDocument>

```

Include dependency graph for simulationengine.hpp:



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



Classes

- class [SimulationEngine](#)

7.25.1 Detailed Description

This file contains the declaration of the [SimulationEngine](#) class.

It is a subclass of the [QGraphicsScene](#) class and represents the simulation engine.

Authors

Tomáš Hobza, Jakub Všeťčka

Date

02.05.2024

Definition in file [simulationengine.hpp](#).

7.26 simulationengine.hpp

[Go to the documentation of this file.](#)

```

00001 /**
00002  * @file simulationengine.hpp
00003  * @brief This file contains the declaration of the SimulationEngine class.
00004  * @details It is a subclass of the QGraphicsScene class and represents the simulation engine.
00005  * @authors Tomáš Hobza, Jakub Všetečka
00006  * @date 02.05.2024
00007  */
00008
00009 #ifndef SIMULATIONENGINE_H
00010 #define SIMULATIONENGINE_H
00011
00012 #include "autorobot.hpp"
00013 #include "gameobject.hpp"
00014 #include "robot.hpp"
00015 #include <QGraphicsScene>
00016 #include <QJsonArray>
00017 #include <QJsonDocument>
00018
00019 class SimulationEngine : public QGraphicsScene {
00020 public:
00021     SimulationEngine(QObject *parent = nullptr, int fps = 60, qreal simulationSpeed = 1.0 / 16.0);
00022
00023     ~SimulationEngine();
00024
00025     /**
00026      * @brief Simulation Frames-Per-Second getter.
00027      * @return int
00028      */
00029     int getFPS();
00030
00031     /**
00032      * @brief Get the time it takes to render a single frame.
00033      * @return int
00034      */
00035     int getFrameTime();
00036
00037     /**
00038      * @brief Set the simulation Frames-Per-Second.
00039      * @param fps
00040      */
00041     void setFPS(int fps);
00042
00043     /**
00044      * @brief Get the simulation speed.
00045      * @return qreal
00046      */
00047     qreal getSimulationSpeed();
00048
00049     /**
00050      * @brief Set the simulation speed.
00051      * @param speed
00052      * @return void
00053      */
00054     void setSimulationSpeed(qreal speed);
00055
00056     /**
00057      * @brief Update the time constant.
00058      * @return void
00059      */
00060     void updateTimeConstant();
00061
00062     /**
00063      * @brief Get the time constant pointer.
00064      * @return qreal*
00065      */
00066     qreal *getTimeConstant();
00067
00068     /**
00069      * @brief Check if a point is inside the scene.
00070      * @param point
00071      * @return bool
00072      */
00073     bool isInsideScene(const QPointF &point) const;
00074
00075     /**
00076      * @brief Get the robot that is currently being controlled.
00077      * @return Robot*
00078      */
00079     Robot *getControlledRobot();
00080
00081     /**
00082      * @brief Set the robot that is currently being controlled.

```

```

00083     * @param robot
00084     * @return void
00085     */
00086 void setControlledRobot(Robot *robot);
00087
00088 /**
00089  * @brief Save the simulation.
00090  * @param filename The name of the file to save the simulation to.
00091  * @details The file will be saved in the JSON format in folder "simulations"
00092  * @return void
00093  */
00094 bool saveSimulation(const QString &filename = "simulation");
00095
00096 /**
00097  * @brief Load the simulation.
00098  * @param filename The name of the file to load the simulation from.
00099  * @details The file will be loaded from the JSON format from folders "simulations" and "exmaples"
00100  * @return void
00101  */
00102 bool loadSimulation(QString filename = "simulation");
00103
00104 /**
00105  * @brief Read the simulation from a JSON object.
00106  * @param json The JSON object to read.
00107  * @return void
00108  */
00109 void read(const QJsonObject &json);
00110
00111 /**
00112  * @brief Convert the simulation to a JSON object.
00113  * @return QJsonObject
00114  */
00115 QJsonObject toJson() const;
00116
00117 /**
00118  * @brief Clear the scene.
00119  *
00120  */
00121 void clearScene();
00122
00123 private:
00124     /** The frames per second of the simulation engine. */
00125     int fps = 60;
00126     /** The speed of the simulation engine. */
00127     qreal simulationSpeed = 1;
00128
00129     /** The time constant of the simulation engine. */
00130     qreal timeConstant = 1;
00131
00132     /** The robot that is currently being controlled. */
00133     Robot *controlledRobot = nullptr;
00134 };
00135
00136 #endif // SIMULATIONENGINE_H

```


Index

- ~AutoRobot
 - AutoRobot, [17](#)
- ~GameObject
 - GameObject, [38](#)
- ~MainWindow
 - MainWindow, [44](#)
- ~Obstacle
 - Obstacle, [54](#)
- ~PopupSaveWindow
 - PopupSaveWindow, [82](#)
- ~Robot
 - Robot, [101](#)
- ~SimulationEngine
 - SimulationEngine, [112](#)
- active
 - Robot, [109](#)
- activeObject
 - OverlayWidget, [63](#)
- anchor
 - OverlayWidget, [61](#)
- angle
 - ParamWidget, [78](#)
- angleToRotate
 - ParamWidget, [78](#)
- AUTO
 - CheckableButton, [25](#)
- autoButton
 - ExpandableButtonWidget, [31](#)
- AutoRobot, [11](#)
 - ~AutoRobot, [17](#)
 - AutoRobot, [17](#)
 - boundingRect, [17](#)
 - collisionLookAhead, [21](#)
 - doRotationStep, [17](#)
 - fromJSON, [18](#)
 - getCollisionLookAhead, [18](#)
 - getRotationDirection, [18](#)
 - getTargetAngle, [18](#)
 - move, [19](#)
 - paint, [19](#)
 - rotationDirection, [21](#)
 - setCollisionLookAhead, [19](#)
 - setRotationDirection, [20](#)
 - setTargetAngle, [20](#)
 - targetAngle, [21](#)
 - toJSON, [20](#)
 - Type, [17](#)
 - type, [20](#)
 - willCollide, [21](#)
- autorobot.hpp, [117](#), [119](#)
 - SMOOTH_ROTATION_SPEED, [119](#)
- BODY_COLLISION_MARGIN
 - robot.hpp, [149](#)
- boundingRect
 - AutoRobot, [17](#)
 - Robot, [102](#)
- CheckableButton, [22](#)
 - AUTO, [25](#)
 - CheckableButton, [25](#)
 - CONT, [25](#)
 - getOverlay, [25](#)
 - getWidgetPos, [25](#)
 - mouseMoveEvent, [26](#)
 - mousePressEvent, [26](#)
 - mouseReleaseEvent, [26](#)
 - ObjectType, [24](#)
 - objType, [27](#)
 - OBST, [25](#)
 - overlay, [27](#)
 - setOverlay, [27](#)
- checkablebutton.hpp, [120](#), [122](#)
- clearScene
 - SimulationEngine, [112](#)
- collapse
 - ExpandableButtonWidget, [30](#)
- collisionLookAhead
 - AutoRobot, [21](#)
- CONT
 - CheckableButton, [25](#)
- controlButton
 - ExpandableButtonWidget, [31](#)
- controlledRobot
 - SimulationEngine, [116](#)
- convertFromRotatedSystem
 - OverlayWidget, [61](#)
- convertToRotatedSystem
 - OverlayWidget, [61](#)
- detectionDistance
 - ParamWidget, [78](#)
- direction
 - ParamWidget, [78](#)
- disconnectStalkedObject
 - ParamWidget, [72](#)
- doRotationStep
 - AutoRobot, [17](#)
- enteredText

- PopupSaveWindow, 83
- eventFilter
 - MainWindow, 44
- expand
 - ExpandableButtonWidget, 31
- ExpandableButtonWidget, 28
 - autoButton, 31
 - collapse, 30
 - controlButton, 31
 - expand, 31
 - ExpandableButtonWidget, 30
 - mainButton, 31
 - obstacleButton, 31
 - setOverlay, 31
- expandableWidget
 - MainWindow, 49
- ExpButton, 32
 - ExpButton, 34
 - mousePressEvent, 35
 - pressed, 35
- expbutton.hpp, 123, 125
- expbuttonwidget.hpp, 125, 127
- focusIn
 - ParamEditLine, 67
 - ParamWidget, 72
- focusInEvent
 - ParamEditLine, 67
- focusOut
 - ParamEditLine, 67
 - ParamWidget, 73
- focusOutEvent
 - ParamEditLine, 67
- fps
 - SimulationEngine, 116
- fromJSON
 - AutoRobot, 18
 - Obstacle, 54
 - Robot, 102
- GameObject, 35
 - ~GameObject, 38
 - GameObject, 38
 - getCenter, 38
 - getPos, 38
 - paint, 38
 - rotation, 39
 - setPos, 39
 - setRotation, 39
 - toJSON, 40
- gameobject.hpp, 127, 129
- getAngle
 - Robot, 102
- getCenter
 - GameObject, 38
 - Obstacle, 55
 - Robot, 102
- getCollisionLookAhead
 - AutoRobot, 18
- getControlledRobot
 - SimulationEngine, 112
- getDirectionVector
 - Robot, 102
- getEnteredText
 - PopupSaveWindow, 83
- getFPS
 - SimulationEngine, 113
- getFrameTime
 - SimulationEngine, 113
- getMoveSpeed
 - Robot, 103
- getOverlay
 - CheckableButton, 25
- getPos
 - GameObject, 38
 - Obstacle, 55
 - Robot, 103
- getRadius
 - Robot, 103
- getRotationDirection
 - AutoRobot, 18
- getRotationSpeed
 - Robot, 103
- getSimulationSpeed
 - SimulationEngine, 113
- getTargetAngle
 - AutoRobot, 18
- getTimeConstant
 - OverlayWidget, 62
 - SimulationEngine, 113
- getWidgetPos
 - CheckableButton, 25
- goLeft
 - MainWindow, 45
- goRight
 - MainWindow, 45
- goStraight
 - MainWindow, 45
- graphView
 - OverlayWidget, 63
- handleItemDoubleClick
 - MainWindow, 45
- hide
 - ParamWidget, 73
- initScene
 - MainWindow, 46
- isActive
 - Robot, 103
- isInsideScene
 - SimulationEngine, 113
- isMoving
 - Robot, 109
- isRotating
 - Robot, 109
- keepUpdating

- ParamWidget, 78
- keyPressEvent
 - Robot, 104
- keyReleaseEvent
 - Robot, 104
- labelAngle
 - ParamWidget, 79
- labelAngleToRotate
 - ParamWidget, 79
- labelDetectionDistance
 - ParamWidget, 79
- labelDirection
 - ParamWidget, 79
- labelRadius
 - ParamWidget, 79
- labelSize
 - ParamWidget, 79
- labelSpeed
 - ParamWidget, 79
- lastMousePos
 - OverlayWidget, 64
- layout
 - ParamWidget, 80
- Left
 - Robot, 101
- lineEdit
 - PopupSaveWindow, 83
- listWidget
 - MainWindow, 49
- loadSimulation
 - SimulationEngine, 114
- mainButton
 - ExpandableButtonWidget, 31
- MainWindow, 40
 - ~MainWindow, 44
 - eventFilter, 44
 - expandableWidget, 49
 - goLeft, 45
 - goRight, 45
 - goStraight, 45
 - handleItemDoubleClick, 45
 - initScene, 46
 - listWidget, 49
 - MainWindow, 44
 - mouseDoubleClickEvent, 46
 - mouseMoveEvent, 46
 - mousePressEvent, 46
 - mouseReleaseEvent, 47
 - on_horizontalSlider_valueChanged, 47
 - on_pushButton_clicked, 47
 - overlay, 49
 - paramWidget, 50
 - resizeEvent, 47
 - saveSimulation, 48
 - setupAnimation, 48
 - showEvent, 48
 - simulationEngine, 50
 - stopMoving, 48
 - stopRotating, 49
 - toggleList, 49
 - ui, 50
 - updateAnimation, 49
- mainwindow.h, 130, 131
- mouseDoubleClickEvent
 - MainWindow, 46
- mouseMoveEvent
 - CheckableButton, 26
 - MainWindow, 46
- mousePressEvent
 - CheckableButton, 26
 - ExpButton, 35
 - MainWindow, 46
- mouseReleaseEvent
 - CheckableButton, 26
 - MainWindow, 47
- move
 - AutoRobot, 19
 - Robot, 104
- move_speed
 - Robot, 109
- navigateTheSea
 - OverlayWidget, 62
- None
 - Robot, 101
- numberValidator
 - ParamWidget, 80
- ObjectType
 - CheckableButton, 24
- objType
 - CheckableButton, 27
- OBST
 - CheckableButton, 25
- Obstacle, 51
 - ~Obstacle, 54
 - fromJSON, 54
 - getCenter, 55
 - getPos, 55
 - Obstacle, 53, 54
 - obstacleSepuku, 55
 - paint, 55
 - paramsUpdated, 56
 - rotation, 56
 - setPos, 56
 - setRotation, 57
 - toJSON, 57
- obstacle.hpp, 133, 135
- obstacleButton
 - ExpandableButtonWidget, 31
- obstacleSepuku
 - Obstacle, 55
- offset
 - OverlayWidget, 64
- on_horizontalSlider_valueChanged
 - MainWindow, 47

- on_pushButton_clicked
 - MainWindow, 47
- onOkClicked
 - PopupSaveWindow, 83
- option
 - OverlayWidget, 64
- overlay
 - CheckableButton, 27
 - MainWindow, 49
- OverlayWidget, 57
 - activeObject, 63
 - anchor, 61
 - convertFromRotatedSystem, 61
 - convertToRotatedSystem, 61
 - getTimeConstant, 62
 - graphView, 63
 - lastMousePos, 64
 - navigateTheSea, 62
 - offset, 64
 - option, 64
 - OverlayWidget, 60
 - paintEvent, 62
 - setActiveObject, 62
 - setLastMousePos, 63
 - simEng, 64
 - trySetSail, 63
- overlaywidget.hpp, 136, 138
- paint
 - AutoRobot, 19
 - GameObject, 38
 - Obstacle, 55
 - Robot, 104
- paintEvent
 - OverlayWidget, 62
- ParamEditLine, 65
 - focusIn, 67
 - focusInEvent, 67
 - focusOut, 67
 - focusOutEvent, 67
 - ParamEditLine, 66
- parameditline.hpp, 139, 141
- paramsUpdated
 - Obstacle, 56
 - Robot, 105
- ParamWidget, 68
 - angle, 78
 - angleToRotate, 78
 - detectionDistance, 78
 - direction, 78
 - disconnectStalkedObject, 72
 - focusIn, 72
 - focusOut, 73
 - hide, 73
 - keepUpdating, 78
 - labelAngle, 79
 - labelAngleToRotate, 79
 - labelDetectionDistance, 79
 - labelDirection, 79
 - labelRadius, 79
 - labelSize, 79
 - labelSpeed, 79
 - layout, 80
 - numberValidator, 80
 - ParamWidget, 72
 - radius, 80
 - setAngle, 73
 - setAngleToRotate, 73
 - setDetectionDistance, 74
 - setDirection, 74
 - setRadius, 74
 - setSize, 74
 - setSpeed, 74
 - setUpEditLine, 75
 - show, 75, 76
 - size, 80
 - speed, 80
 - stalk, 76, 77
 - stalkedObject, 80
 - stopStalking, 77
 - updateAutoRobot, 77
 - updateObstacle, 77
 - updateRobot, 78
- paramWidget
 - MainWindow, 50
- paramwidget.hpp, 141, 143
- PopupSaveWindow, 81
 - ~PopupSaveWindow, 82
 - enteredText, 83
 - getEnteredText, 83
 - lineEdit, 83
 - onOkClicked, 83
 - PopupSaveWindow, 82
- popupsavewindow.h, 145, 146
- pos
 - Robot, 105
- pressed
 - ExpButton, 35
- QDialog, 84
- QGraphicsEllipseItem, 85
- QGraphicsRectItem, 86
- QGraphicsScene, 88
- QLineEdit, 89
- QMainWindow, 90
- QObject, 92
- QPushButton, 93
- QWidget, 95
- radius
 - ParamWidget, 80
- read
 - SimulationEngine, 114
- resizeEvent
 - MainWindow, 47
- Right
 - Robot, 101
- Robot, 96

- ~Robot, 101
- active, 109
- boundingRect, 102
- fromJSON, 102
- getAngle, 102
- getCenter, 102
- getDirectionVector, 102
- getMoveSpeed, 103
- getPos, 103
- getRadius, 103
- getRotationSpeed, 103
- isActive, 103
- isMoving, 109
- isRotating, 109
- keyPressEvent, 104
- keyReleaseEvent, 104
- Left, 101
- move, 104
- move_speed, 109
- None, 101
- paint, 104
- paramsUpdated, 105
- pos, 105
- Right, 101
- Robot, 101
- robotSepuku, 105
- rotation, 105
- rotation_speed, 109
- RotationDirection, 101
- setMoveSpeed, 105
- setPos, 106
- setRadius, 106
- setRotation, 106
- setRotationSpeed, 107
- startMoving, 107
- startRotating, 107
- stopMoving, 107
- stopRotating, 107
- timeConstant, 110
- toggleActive, 107
- toJSON, 108
- Type, 100
- type, 108
- willCollide, 108
- robot.hpp, 147, 149
 - BODY_COLLISION_MARGIN, 149
- robotSepuku
 - Robot, 105
- rotation
 - GameObject, 39
 - Obstacle, 56
 - Robot, 105
- rotation_speed
 - Robot, 109
- RotationDirection
 - Robot, 101
- rotationDirection
 - AutoRobot, 21
- saveSimulation
 - MainWindow, 48
 - SimulationEngine, 114
- setActiveObject
 - OverlayWidget, 62
- setAngle
 - ParamWidget, 73
- setAngleToRotate
 - ParamWidget, 73
- setCollisionLookAhead
 - AutoRobot, 19
- setControlledRobot
 - SimulationEngine, 115
- setDetectionDistance
 - ParamWidget, 74
- setDirection
 - ParamWidget, 74
- setFPS
 - SimulationEngine, 115
- setLastMousePos
 - OverlayWidget, 63
- setMoveSpeed
 - Robot, 105
- setOverlay
 - CheckableButton, 27
 - ExpandableButtonWidget, 31
- setPos
 - GameObject, 39
 - Obstacle, 56
 - Robot, 106
- setRadius
 - ParamWidget, 74
 - Robot, 106
- setRotation
 - GameObject, 39
 - Obstacle, 57
 - Robot, 106
- setRotationDirection
 - AutoRobot, 20
- setRotationSpeed
 - Robot, 107
- setSimulationSpeed
 - SimulationEngine, 115
- setSize
 - ParamWidget, 74
- setSpeed
 - ParamWidget, 74
- setTargetAngle
 - AutoRobot, 20
- setupAnimation
 - MainWindow, 48
- setUpEditLine
 - ParamWidget, 75
- show
 - ParamWidget, 75, 76
- showEvent
 - MainWindow, 48
- simEng

- OverlayWidget, 64
- SimulationEngine, 110
 - ~SimulationEngine, 112
 - clearScene, 112
 - controlledRobot, 116
 - fps, 116
 - getControlledRobot, 112
 - getFPS, 113
 - getFrameTime, 113
 - getSimulationSpeed, 113
 - getTimeConstant, 113
 - isInsideScene, 113
 - loadSimulation, 114
 - read, 114
 - saveSimulation, 114
 - setControlledRobot, 115
 - setFPS, 115
 - setSimulationSpeed, 115
 - SimulationEngine, 112
 - simulationSpeed, 116
 - timeConstant, 116
 - toJson, 116
 - updateTimeConstant, 116
- simulationEngine
 - MainWindow, 50
- simulationengine.hpp, 152, 154
- simulationSpeed
 - SimulationEngine, 116
- size
 - ParamWidget, 80
- SMOOTH_ROTATION_SPEED
 - autorobot.hpp, 119
- speed
 - ParamWidget, 80
- stalk
 - ParamWidget, 76, 77
- stalkedObject
 - ParamWidget, 80
- startMoving
 - Robot, 107
- startRotating
 - Robot, 107
- stopMoving
 - MainWindow, 48
 - Robot, 107
- stopRotating
 - MainWindow, 49
 - Robot, 107
- stopStalking
 - ParamWidget, 77
- targetAngle
 - AutoRobot, 21
- timeConstant
 - Robot, 110
 - SimulationEngine, 116
- toggleActive
 - Robot, 107
- toggleList
 - MainWindow, 49
- toJson
 - AutoRobot, 20
 - GameObject, 40
 - Obstacle, 57
 - Robot, 108
- toJson
 - SimulationEngine, 116
- trySetSail
 - OverlayWidget, 63
- Type
 - AutoRobot, 17
 - Robot, 100
- type
 - AutoRobot, 20
 - Robot, 108
- Ui, 9
- ui
 - MainWindow, 50
- updateAnimation
 - MainWindow, 49
- updateAutoRobot
 - ParamWidget, 77
- updateObstacle
 - ParamWidget, 77
- updateRobot
 - ParamWidget, 78
- updateTimeConstant
 - SimulationEngine, 116
- willCollide
 - AutoRobot, 21
 - Robot, 108