

RTM

0.1.0

Создано системой Doxygen 1.8.13



# Оглавление

1	Титульная страница	1
2	Список задач	3
3	Алфавитный указатель пространств имен	5
3.1	Пространства имен . . . . .	5
4	Иерархический список классов	7
4.1	Иерархия классов . . . . .	7
5	Алфавитный указатель классов	9
5.1	Классы . . . . .	9
6	Пространства имен	11
6.1	Пространство имен <code>rtm</code> . . . . .	11
6.1.1	Подробное описание . . . . .	17
6.1.2	Типы . . . . .	17
6.1.2.1	<code>Directions</code> . . . . .	17
6.1.2.2	<code>LinesCounts</code> . . . . .	17
6.1.2.3	<code>DirectionSignals</code> . . . . .	17
6.1.2.4	<code>CrossroadSignals</code> . . . . .	18
6.1.2.5	<code>SignalSprites</code> . . . . .	18
6.1.2.6	<code>SignalsSprites</code> . . . . .	18
6.1.2.7	<code>DirectionsSignalSprites</code> . . . . .	18
6.1.3	Перечисления . . . . .	18
6.1.3.1	<code>AngleType</code> . . . . .	18

6.1.3.2	DirectionType	19
6.1.3.3	CoatingUnionType	19
6.1.3.4	DirectionSignalIndex	20
6.1.3.5	SignalType	20
6.1.3.6	StateType	20
6.1.3.7	CoatingType	21
6.1.3.8	SignalFileId	21
6.1.4	Функции	21
6.1.4.1	CheckCollisions()	21
6.1.4.2	SameCoordinates()	21
6.1.4.3	RoundCoordinate()	22
6.1.4.4	RoundToCenter()	22
6.1.4.5	InCenter()	23
6.1.4.6	DistanceToNextCenter()	23
6.1.4.7	CenterIsCrossed()	23
6.1.4.8	SameAngles()	24
6.1.4.9	RoundAngle()	24
6.1.4.10	NormalizeAngle()	25
6.1.4.11	PixelToCell()	25
6.1.4.12	CellToPixel()	26
6.1.4.13	AngleToAngleType()	26
6.1.4.14	AngleToDirection()	26
6.1.4.15	AngleTypeToAngle()	27
6.1.4.16	AngleTypeToDirection()	27
6.1.4.17	DirectionToAngle()	27
6.1.4.18	DirectionToAngleType()	28
6.1.4.19	SumAngleTypes()	28
6.1.4.20	CountDeceleration()	29
6.1.5	Переменные	29
6.1.5.1	NEAR_DELTA	29
6.1.5.2	DEFAULT_CROSSROAD_SIGNALS	29
6.1.5.3	DEFAULT_DIRECTIONS_SIGNAL_SPRITES	30
6.1.5.4	ROADS_RESISTANCES	30
6.1.5.5	ROADS_DIRECTIONS	30
6.1.5.6	CARS_MAX_SPEEDS	31
6.1.5.7	CARS_ACCELERATIONS	31

7	Классы	33
7.1	Класс AppDelegate	33
7.1.1	Подробное описание	34
7.1.2	Методы	34
7.1.2.1	applicationDidFinishLaunching()	34
7.2	Класс rtm::BuildingObject	34
7.2.1	Подробное описание	35
7.2.2	Конструктор(ы)	35
7.2.2.1	BuildingObject() [1/3]	35
7.2.2.2	BuildingObject() [2/3]	35
7.2.2.3	BuildingObject() [3/3]	36
7.2.3	Методы	36
7.2.3.1	GetClassFile_()	36
7.3	Класс rtm::CarObject	37
7.3.1	Подробное описание	38
7.3.2	Конструктор(ы)	38
7.3.2.1	CarObject() [1/3]	39
7.3.2.2	CarObject() [2/3]	39
7.3.2.3	CarObject() [3/3]	39
7.3.3	Методы	40
7.3.3.1	MovementStart_()	40
7.3.3.2	MovementTick_()	40
7.3.3.3	MovementEnd_()	41
7.3.3.4	LineChangingStart()	41
7.3.3.5	CheckCoatingAhead_()	41
7.3.3.6	CheckCoatingUnionAhead_()	41
7.3.3.7	CheckRoadAhead_()	42
7.3.3.8	GetClassFile_()	42
7.3.3.9	GetClassMaxSpeed_()	42
7.3.3.10	GetClassAcceleration_()	43

7.4	Класс <code>rtm::CoatingObject</code> . . . . .	43
7.4.1	Подробное описание . . . . .	44
7.4.2	Конструктор(ы) . . . . .	44
7.4.2.1	<code>CoatingObject()</code> [1/2] . . . . .	45
7.4.2.2	<code>CoatingObject()</code> [2/2] . . . . .	45
7.4.3	Методы . . . . .	45
7.4.3.1	<code>GetSprite()</code> . . . . .	46
7.4.3.2	<code>GetResistance()</code> . . . . .	46
7.4.3.3	<code>HasDirection()</code> . . . . .	46
7.4.3.4	<code>IsDirectionAvailable()</code> . . . . .	46
7.4.3.5	<code>SetDirectionAvailability()</code> . . . . .	47
7.4.3.6	<code>SetSprite_()</code> . . . . .	47
7.4.3.7	<code>SetX_()</code> . . . . .	47
7.4.3.8	<code>SetY_()</code> . . . . .	48
7.5	Класс <code>rtm::CoatingUnion</code> . . . . .	48
7.5.1	Подробное описание . . . . .	49
7.5.2	Конструктор(ы) . . . . .	49
7.5.2.1	<code>CoatingUnion()</code> . . . . .	49
7.5.3	Методы . . . . .	50
7.5.3.1	<code>GetType()</code> . . . . .	50
7.5.3.2	<code>GetWidth()</code> . . . . .	50
7.5.3.3	<code>GetHeight()</code> . . . . .	50
7.5.3.4	<code>GetCoatingObject()</code> . . . . .	50
7.5.3.5	<code>GetLength()</code> . . . . .	51
7.5.3.6	<code>IsCorrectColumn()</code> . . . . .	51
7.5.3.7	<code>IsCorrectRow()</code> . . . . .	51
7.5.3.8	<code>ShowSprites()</code> . . . . .	51
7.5.3.9	<code>ReleaseSprites()</code> . . . . .	52
7.5.3.10	<code>GetColumn_()</code> . . . . .	52
7.5.3.11	<code>GetRow_()</code> . . . . .	52

7.6	Класс <code>rtm::ControlUnit</code> . . . . .	53
7.6.1	Подробное описание . . . . .	54
7.6.2	Конструктор(ы) . . . . .	54
7.6.2.1	<code>ControlUnit()</code> [1/2] . . . . .	54
7.6.2.2	<code>ControlUnit()</code> [2/2] . . . . .	54
7.6.3	Методы . . . . .	55
7.6.3.1	<code>Update()</code> . . . . .	55
7.6.3.2	<code>operator bool()</code> . . . . .	55
7.6.3.3	<code>GetSignal()</code> . . . . .	55
7.6.3.4	<code>ShowSprites()</code> . . . . .	56
7.6.3.5	<code>ReleaseSprites()</code> . . . . .	56
7.6.3.6	<code>UpdateSignal_()</code> . . . . .	56
7.6.3.7	<code>SetState_()</code> . . . . .	57
7.6.3.8	<code>GetSignalFile_()</code> . . . . .	57
7.7	Класс <code>rtm::CrossroadObject</code> . . . . .	57
7.7.1	Подробное описание . . . . .	58
7.7.2	Конструктор(ы) . . . . .	59
7.7.2.1	<code>CrossroadObject()</code> [1/2] . . . . .	59
7.7.2.2	<code>CrossroadObject()</code> [2/2] . . . . .	59
7.7.3	Методы . . . . .	59
7.7.3.1	<code>CrossroadMatrix()</code> . . . . .	60
7.7.3.2	<code>TCrossroadMatrix()</code> . . . . .	60
7.7.3.3	<code>GetNullDirection()</code> . . . . .	61
7.7.3.4	<code>GetControlUnit()</code> . . . . .	61
7.7.3.5	<code>ShowSprites()</code> . . . . .	61
7.7.3.6	<code>ReleaseSprites()</code> . . . . .	61
7.8	Класс <code>rtm::DrivewayObject</code> . . . . .	62
7.8.1	Подробное описание . . . . .	63
7.8.2	Конструктор(ы) . . . . .	63
7.8.2.1	<code>DrivewayObject()</code> . . . . .	63

7.8.3	Методы . . . . .	64
7.8.3.1	DrivewayMatrix() . . . . .	64
7.8.3.2	GetLength() . . . . .	64
7.8.3.3	GetLinesCount() . . . . .	65
7.8.3.4	isRightLine() [1/2] . . . . .	65
7.8.3.5	isRightLine() [2/2] . . . . .	65
7.8.3.6	isLeftLine() [1/2] . . . . .	66
7.8.3.7	isLeftLine() [2/2] . . . . .	66
7.8.3.8	CountLength_() . . . . .	66
7.8.3.9	CountLines_() . . . . .	67
7.9	Класс rtm::DynamicObject . . . . .	67
7.9.1	Подробное описание . . . . .	69
7.9.2	Конструктор(ы) . . . . .	69
7.9.2.1	DynamicObject() [1/2] . . . . .	69
7.9.2.2	DynamicObject() [2/2] . . . . .	69
7.9.3	Методы . . . . .	70
7.9.3.1	GetSpeed() . . . . .	70
7.9.3.2	GetLastDelta() . . . . .	70
7.9.3.3	HasCollision() . . . . .	71
7.9.3.4	Update() . . . . .	71
7.9.3.5	IsNearOthers() . . . . .	71
7.9.3.6	SetSpeed_() . . . . .	71
7.9.3.7	SetCollisionFlag_() . . . . .	72
7.9.3.8	IsBeholding_() . . . . .	72
7.9.3.9	IsIntersecting_() . . . . .	72
7.9.3.10	IsNear_() . . . . .	73
7.9.4	Документация по друзьям класса и функциям, относящимся к классу . . . . .	73
7.9.4.1	CheckCollisions . . . . .	73
7.10	Класс rtm::MapObject . . . . .	74
7.10.1	Подробное описание . . . . .	74



7.10.2	Конструктор(ы)	74
7.10.2.1	MapObject() [1/2]	75
7.10.2.2	MapObject() [2/2]	75
7.11	Класс rtm::RoadCoating	75
7.11.1	Подробное описание	76
7.11.2	Конструктор(ы)	76
7.11.2.1	RoadCoating() [1/3]	76
7.11.2.2	RoadCoating() [2/3]	77
7.11.2.3	RoadCoating() [3/3]	77
7.11.3	Методы	78
7.11.3.1	GetClassFile_()	78
7.11.3.2	GetClassResistance_()	78
7.11.3.3	GetClassDirections_()	79
7.12	Структура rtm::SpawnType	79
7.12.1	Подробное описание	79
7.13	Класс rtm::StaticObject	80
7.13.1	Подробное описание	80
7.13.2	Конструктор(ы)	80
7.13.2.1	StaticObject() [1/2]	81
7.13.2.2	StaticObject() [2/2]	81
7.14	Класс rtm::TurnObject	81
7.14.1	Подробное описание	82
7.14.2	Конструктор(ы)	83
7.14.2.1	TurnObject()	83
7.14.3	Методы	83
7.14.3.1	RightTurnMatrix()	83
7.14.3.2	LeftTurnMatrix()	84
7.14.3.3	IsRight()	84
7.14.3.4	GetAngle()	84
7.15	Класс rtm::VehicleObject	85

7.15.1	Подробное описание . . . . .	87
7.15.2	Конструктор(ы) . . . . .	87
7.15.2.1	VehicleObject() [1/2] . . . . .	87
7.15.2.2	VehicleObject() [2/2] . . . . .	88
7.15.3	Методы . . . . .	88
7.15.3.1	Update() . . . . .	88
7.15.3.2	MoveForward_() . . . . .	88
7.15.3.3	Stop_() . . . . .	89
7.15.3.4	Rotate_() . . . . .	89
7.15.3.5	ChangeLine_() . . . . .	89
7.15.3.6	IsMovement_() . . . . .	90
7.15.3.7	IsRotation_() . . . . .	90
7.15.3.8	IsLineChanging_() . . . . .	90
7.15.3.9	IsBraking_() . . . . .	90
7.15.3.10	GetMaxSpeed_() . . . . .	91
7.15.3.11	GetFinalSpeed_() . . . . .	91
7.15.3.12	SetFinalSpeed_() . . . . .	91
7.15.3.13	SetBrakingFactor_() . . . . .	91
7.15.3.14	StopAtDistance_() . . . . .	92
7.15.3.15	CheckForwardCoating_() . . . . .	92
7.15.3.16	CheckForwardCoatingUnion_() . . . . .	92
7.15.3.17	CheckForwardArea_() . . . . .	93
7.15.3.18	CheckMovingArea_() . . . . .	93
7.15.3.19	CheckTurnArea_() . . . . .	94
7.15.3.20	CheckRotationArea_() . . . . .	94
7.15.3.21	CheckCrossroadArea_() . . . . .	94
7.15.3.22	CheckLineChangingArea_() . . . . .	95
7.15.3.23	BeforeMoving_() . . . . .	95
7.15.3.24	AfterMoving_() . . . . .	95
7.15.3.25	MovementStart_() . . . . .	96

7.15.3.26 MovementTick_()	96
7.15.3.27 MovementEnd_()	96
7.15.3.28 RotationStart_()	97
7.15.3.29 RotationTick_()	97
7.15.3.30 RotationEnd_()	98
7.15.3.31 LineChangingStart()	98
7.15.3.32 LineChangingTick_()	98
7.15.3.33 LineChangingEnd_()	99
7.15.3.34 LineChanging_()	99
7.15.3.35 Rotation_()	99
7.15.3.36 Movement_()	100
7.15.3.37 SpeedChanging_()	100
7.15.3.38 SmoothBrakingCounter()	100
7.16 Класс rtm::WorldController	100
7.16.1 Подробное описание	103
7.16.2 Конструктор(ы)	103
7.16.2.1 WorldController() [1/3]	103
7.16.2.2 WorldController() [2/3]	103
7.16.2.3 WorldController() [3/3]	104
7.16.3 Методы	104
7.16.3.1 Update()	104
7.16.3.2 GetLayer()	104
7.16.3.3 GetColumnsCount()	105
7.16.3.4 GetRowsCount()	105
7.16.3.5 GetDeltaTime()	105
7.16.3.6 GetTimeFactor()	105
7.16.3.7 GetCoatingObject()	106
7.16.3.8 GetCoatingUnion()	106
7.16.3.9 GetStaticObject()	106
7.16.3.10 GetDynamicObjects()	106

7.16.3.11 IsPause()	107
7.16.3.12 IsCorrectColumn()	107
7.16.3.13 IsCorrectRow()	107
7.16.3.14 IsAllowableColumn()	108
7.16.3.15 IsAllowableRow()	108
7.16.3.16 IsVisibleColumn()	108
7.16.3.17 IsVisibleRow()	109
7.16.3.18 SetTimeFactor()	109
7.16.3.19 LoadMap() [1/2]	109
7.16.3.20 LoadMap() [2/2]	110
7.16.3.21 IsEmpty_()	110
7.16.3.22 GenerateObject_()	110
7.16.3.23 AddCoatingUnion_()	111
7.16.3.24 AddDriveway_()	111
7.16.3.25 AddCrossroad_()	112
7.16.3.26 AddTCrossroad_()	112
7.16.3.27 AddLeftTurt_()	113
7.16.3.28 AddRightTurt_()	113
7.16.3.29 AddControlUnit_()	114
7.16.3.30 AddStaticObject_()	114
7.16.3.31 AddBuilding_()	114
7.16.3.32 AddDynamicObject_()	115
7.16.3.33 AddCar_()	115
7.16.3.34 GetVectorColumn_()	116
7.16.3.35 GetVectorRow_()	116
7.16.3.36 GetRealColumn_()	117
7.16.3.37 GetRealRow_()	117
7.16.3.38 GetClassFile_()	117
7.17 Класс rtm::WorldObject	118
7.17.1 Подробное описание	120

7.17.2	Конструктор(ы)	120
7.17.2.1	WorldObject() [1/2]	120
7.17.2.2	WorldObject() [2/2]	120
7.17.3	Методы	120
7.17.3.1	GetSprite()	121
7.17.3.2	GetX_()	121
7.17.3.3	GetY_()	121
7.17.3.4	GetAngle()	121
7.17.3.5	GetWidth()	122
7.17.3.6	GetHeight()	122
7.17.3.7	SetSprite_()	122
7.17.3.8	SetX_()	122
7.17.3.9	SetY_()	123
7.17.3.10	SetAngle_()	123
7.17.3.11	SetWidth_()	123
7.17.3.12	SetHeight_()	123
7.17.3.13	SetSpriteX_()	124
7.17.3.14	SetSpriteY_()	124
7.17.3.15	SetSpriteAngle_()	124
7.17.3.16	SetSpriteWidth_()	125
7.17.3.17	SetSpriteHeight_()	125
7.18	Класс rtm::WorldScene	125
7.18.1	Подробное описание	127
7.18.2	Методы	127
7.18.2.1	Create()	128
7.18.2.2	init()	128
7.18.2.3	update()	128
7.18.2.4	GetMainLayer()	128
7.18.2.5	SetBackground() [1/2]	129
7.18.2.6	SetBackground() [2/2]	129
7.18.2.7	GetMap_()	129
7.18.2.8	GetBackgroundFile_()	129



# Глава 1

## Титульная страница

Этот проект представляет из себя систему для моделирования дорожного движения. В нём можно тестировать системы управления светофорами, машинами (в ближайшем будущем это понадобится) и многое другое. А также можно просто позалипать на машинки, интересно катающиеся по дорогами, которые может создать любой человек! с:

Версия

0.1.0

Автор

Владимир Северов (Vladimir Severov)

**Необходимо сделать** Предстоит ещё много работы для серьезного использования данной системы, однако в рамках курсового проекта этот проект можно считать успешным.





## Глава 2

### Список задач

page [Титульная страница](#)

Предстоит ещё много работы для серьезного использования данной системы, однако в рамках курсового проекта этот проект можно считать успешным.



## Глава 3

# Алфавитный указатель пространств имен

### 3.1 Пространства имен

Полный список документированных пространств имен.

<a href="#">rtm</a>	Пространство имен для проекта RTM . . . . .	<a href="#">11</a>
---------------------	---	--------------------



## Глава 4

# Иерархический список классов

### 4.1 Иерархия классов

Иерархия классов.

Application	
AppDelegate . . . . .	33
rtm::CoatingObject . . . . .	43
rtm::RoadCoating . . . . .	75
rtm::CoatingUnion . . . . .	48
rtm::CrossroadObject . . . . .	57
rtm::DrivewayObject . . . . .	62
rtm::TurnObject . . . . .	81
rtm::ControlUnit . . . . .	53
Scene	
rtm::WorldScene . . . . .	125
rtm::SpawnType . . . . .	79
rtm::WorldController . . . . .	100
rtm::WorldObject . . . . .	118
rtm::DynamicObject . . . . .	67
rtm::VehicleObject . . . . .	85
rtm::CarObject . . . . .	37
rtm::StaticObject . . . . .	80
rtm::MapObject . . . . .	74
rtm::BuildingObject . . . . .	34



## Глава 5

# Алфавитный указатель классов

### 5.1 Классы

Классы с их кратким описанием.

<a href="#">AppDelegate</a>	Приложение, основанное на Cocos2d . . . . .	33
<a href="#">rtm::BuildingObject</a>	Класс, описывающий строения (здания) . . . . .	34
<a href="#">rtm::CarObject</a>	Класс, описывающий машины . . . . .	37
<a href="#">rtm::CoatingObject</a>	Класс покрытия секции карты . . . . .	43
<a href="#">rtm::CoatingUnion</a>	. . . . .	48
<a href="#">rtm::ControlUnit</a>	Класс управляющего блока (светофор) . . . . .	53
<a href="#">rtm::CrossroadObject</a>	Класс пересечения дорог . . . . .	57
<a href="#">rtm::DrivewayObject</a>	Класс прямой дороги . . . . .	62
<a href="#">rtm::DynamicObject</a>	Класс динамического объекта (который двигается, обновляется) . . . . .	67
<a href="#">rtm::MapObject</a>	Класс статического объекта карты . . . . .	74
<a href="#">rtm::RoadCoating</a>	Класс, описывающий дороги . . . . .	75
<a href="#">rtm::SpawnType</a>	Структура, описывающая параметры точки генерации объектов . . . . .	79
<a href="#">rtm::StaticObject</a>	Класс статического объекта (который не обновляется) . . . . .	80
<a href="#">rtm::TurnObject</a>	Класс поворота дороги . . . . .	81
<a href="#">rtm::VehicleObject</a>	Класс транспорта (динамического объекта карты) . . . . .	85
<a href="#">rtm::WorldController</a>	Класс контроллера карты, связующее звено всех объектов . . . . .	100
<a href="#">rtm::WorldObject</a>	Класс объекта мира (родитель всех условно объемных объектов) . . . . .	118
<a href="#">rtm::WorldScene</a>	Класс главной сцены, на которой всё и происходит (для отрисовки) . . . . .	125





## Глава 6

# Пространства имен

### 6.1 Пространство имен rtm

Пространство имен для проекта RTM.

Классы

- class [BuildingObject](#)  
Класс, описывающий строения (здания)
- class [CarObject](#)  
Класс, описывающий машины
- class [CoatingObject](#)  
Класс покрытия секции карты
- class [CoatingUnion](#)
- class [ControlUnit](#)  
Класс управляющего блока (светофор)
- class [CrossroadObject](#)  
Класс пересечения дорог
- class [DrivewayObject](#)  
Класс прямой дороги
- class [DynamicObject](#)  
Класс динамического объекта (который движется, обновляется)
- class [MapObject](#)  
Класс статического объекта карты
- class [RoadCoating](#)  
Класс, описывающий дороги
- struct [SpawnType](#)  
Структура, описывающая параметры точки генерации объектов
- class [StaticObject](#)  
Класс статического объекта (который не обновляется)
- class [TurnObject](#)  
Класс поворота дороги
- class [VehicleObject](#)  
Класс транспорта (динамического объекта карты)
- class [WorldController](#)  
Класс контроллера карты, связующее звено всех объектов
- class [WorldObject](#)  
Класс объекта мира (родитель всех условно объемных объектов)
- class [WorldScene](#)  
Класс главной сцены, на которой всё и происходит (для отрисовки)

## Определения типов

- using `WorldControllerUnique` = `std::unique_ptr< WorldController >`  
Умный указатель для класса `WorldController`.
- using `SpawnVector` = `std::vector< SpawnType >`  
Массив точек генерации объектов
- using `CoatingUnique` = `std::unique_ptr< CoatingObject >`  
Умный указатель для класса `CoatingObject`.
- using `CoatingVector` = `std::vector< CoatingUnique >`  
Массив объектов класса `CoatingObject`.
- using `CoatingMatrix` = `std::vector< CoatingVector >`  
Матрица объектов класса `CoatingObject`.
- using `CoatingUnionShared` = `std::shared_ptr< CoatingUnion >`  
Умный указатель для класса `CoatingUnion`.
- using `CoatingUnionVector` = `std::vector< CoatingUnionShared >`  
Массив объектов класса `CoatingUnion`.
- using `CoatingUnionMatrix` = `std::vector< CoatingUnionVector >`  
Матрица объектов класса `CoatingUnion`.
- using `ControlUnitShared` = `std::shared_ptr< ControlUnit >`  
Умный указатель для класса `ControlUnit`.
- using `ControlUnitVector` = `std::vector< ControlUnitShared >`  
Массив объектов класса `ControlUnit`.
- using `StaticShared` = `std::shared_ptr< StaticObject >`  
Умный указатель для класса `StaticObject`.
- using `StaticVector` = `std::vector< StaticShared >`  
Массив объектов класса `StaticObject`.
- using `StaticMatrix` = `std::vector< StaticVector >`  
Матрица объектов класса `StaticObject`.
- using `DynamicShared` = `std::shared_ptr< DynamicObject >`  
Умный указатель для класса `DynamicObject`.
- using `DynamicVector` = `std::vector< DynamicShared >`  
Массив объектов класса `DynamicObject`.
- using `Directions` = `std::array< bool, 8 >`
- using `LinesCounts` = `std::array< size_t, 4 >`
- using `DirectionSignals` = `std::array< SignalType, 4 >`
- using `CrossroadSignals` = `std::array< DirectionSignals, 4 >`
- using `SignalSprites` = `std::array< cocos2d::Sprite *, 5 >`
- using `SignalsSprites` = `std::array< SignalSprites, 3 >`
- using `DirectionsSignalSprites` = `std::array< SignalsSprites, 4 >`

## Перечисления

- enum `AngleType` {  
  `NullAngle` = -1, `Up` = 0, `Right`, `Down`,  
  `Left`, `UpRight`, `DownRight`, `DownLeft`,  
  `UpLeft` }  
Тип для определения положения некоторых объектов и индикации разрешенных направлений на кусочке объекта
- enum `DirectionType` {  
  `NullDirection` = -1, `Upward` = 0, `Rightward`, `Downward`,  
  `Leftward` }  
Тип для задания направления движения транспорта

- enum `CoatingUnionType` {  
`NoCoatingUnion` = -1, `DrivewayType`, `CrossroadType`, `TCrossroadType`,  
`TurnType` }  
 Возможные типы дорожных объединений
- enum `DirectionSignalIndex` { `ForwardSignalIndex` = 0, `LeftwardSignalIndex`, `RightwardSignalIndex` }  
 Индексы массива для каждого типа сигнала
- enum `SignalType` {  
`NotWorking` = 0, `Allowed`, `Warning`, `Forbidden`,  
`Closed` }  
 Возможные сигналы светофора
- enum `StateType` { `NotStarted`, `MustStart`, `Started`, `MustStop` }  
 Возможные состояния для манёвров (движение, поворот, перестроение)
- enum `CoatingType` { `AsphaltCoating` = 0 }  
 Индексы, начиная с которых начинаются текстуры покрытий определенного типа
- enum `SignalFileId` { `ForwardSignalId` = 1, `LeftwardSignalId` = 6, `RightwardSignalId` = 11 }  
 Индексы, начиная с которых начинаются текстуры сигнала определенного типа

## Функции

- void `CheckCollisions` (`WorldController` \*const world)
- `AngleType` `SumAngleTypes` (`AngleType` a, `AngleType` b)
- float `CountDeceleration` (float maxSpeed)

## Функции для работы с параметрами положения объектов

- bool `SameCoordinates` (float a, float b, float delta=`COORD_DELTA`)
- float `RoundCoordinate` (float coordinate, float delta=`COORD_DELTA`)
- float `RoundToCenter` (float coordinate)
- bool `InCenter` (float coordinate, float delta=`COORD_DELTA`)
- float `DistanceToNextCenter` (float x, float y, float angle)
- bool `CenterIsCrossed` (float x, float y, float angle, float lastDelta)
- bool `SameAngles` (float a, float b, float delta=`ANGLE_DELTA`)
- float `RoundAngle` (float angle, float delta=`ANGLE_DELTA`)
- float `NormalizeAngle` (float angle)

## Конверторы схожих типов

- int `PixelToCell` (float coordinate)
- float `CellToPixel` (int cellNumber)
- `AngleType` `AngleToAngleType` (float angle)
- `DirectionType` `AngleToDirection` (float angle)
- float `AngleTypeToAngle` (`AngleType` angle)
- `DirectionType` `AngleTypeToDirection` (`AngleType` angle)
- float `DirectionToAngle` (`DirectionType` direction)
- `AngleType` `DirectionToAngleType` (`DirectionType` direction)

## Переменные

Константы для флага isRight

- bool const `LEFT` { false }  
Влево
- bool const `RIGHT` { true }  
Вправо

Заранее посчитанные операции над  $\pi$

- float const `F_PI_8` { 0.392699081698724154808f }  
 $\pi / 8$
- float const `F_PI_4` { 0.785398163397448309616f }  
 $\pi / 4$
- float const `F_PI_2` { 1.57079632679489661923f }  
 $\pi / 2$
- float const `F_PI` { 3.14159265358979323846f }  
 $\pi$
- float const `F_2_PI` { 6.28318530717958647692f }  
 $2 * \pi$

Константы для конвертации углов из радиан в градусы и обратно

- float const `DEG_RAD` { `F_PI` / 180.f }  
Коэффициент для перевода из градусов в радианы
- float const `RAD_DEG` { 180.f / `F_PI` }  
Коэффициент для перевода из радиан в градусы

Заранее посчитанные углы

- float const `ANGLE_UP` { 0.f }  
Угол вверх
- float const `ANGLE_RIGHT` { `F_PI_2` }  
Угол вправо
- float const `ANGLE_DOWN` { -`F_PI` }  
Угол вниз
- float const `ANGLE_LEFT` { -`F_PI_2` }  
Угол влево
- float const `ANGLE_UP_RIGHT` { `F_PI_4` }  
Угол по диагонали вверх вправо
- float const `ANGLE_DOWN_RIGHT` { `F_PI` - `F_PI_4` }  
Угол по диагонали вниз вправо
- float const `ANGLE_DOWN_LEFT` { -`F_PI` + `F_PI_4` }  
Угол по диагонали вниз влево
- float const `ANGLE_UP_LEFT` { -`F_PI_4` }  
Угол по диагонали вверх влево

Допустимые погрешности

- float const `ANGLE_DELTA` { 1.f \* `DEG_RAD` }  
Погрешность для углов
- float const `COORD_DELTA` { 1.f }  
Погрешность для координат
- float const `NEAR_DELTA` { 1.f }

### Параметры карт

- `size_t const CELL_SIZE { 30 }`  
Длина (ширина) ячейки карты
- `size_t const ROTATION_RADIUS { CELL_SIZE }`  
Желаемый радиус поворота транспорта
- `float const MIN_TIME_FACTOR { 0.5f }`  
Минимальный коэффициент ускорения времени. Если меньше 1, то замедление
- `float const MAX_TIME_FACTOR { 4.f }`  
Максимальный коэффициент ускорения времени

Номера слоев для разных объектов. Чем больше, тем выше (ближе к нам)

- `int const BACKGROUND_Z_ORDER { -1 }`  
Номер слоя для слоя фона
- `int const MAIN_Z_ORDER { 0 }`  
Номер слоя для главного слоя (на нем все объекты)
- `int const COATING_OBJECT_Z_ORDER { 1 }`  
Номер слоя для покрытий (дорог)
- `int const LEFTWARD_SIGNAL_Z_ORDER { 2 }`  
Номер слоя для левых стрелок светофора
- `int const RIGHTWARD_SIGNAL_Z_ORDER { 3 }`  
Номер слоя для правых стрелок светофора
- `int const FORWARD_SIGNAL_Z_ORDER { 4 }`  
Номер слоя для светофора в прямом направлении
- `int const VEHICLE_OBJECT_Z_ORDER { 5 }`  
Номер слоя для транспорта
- `int const MAP_OBJECT_Z_ORDER { 6 }`  
Номер слоя для статичных объектов карты

Область видимости при движении вперед

- `float const VIEW_RADIUS { 60.f }`  
Радиус
- `float const VIEW_ANGLE { 24.f * DEG_RAD }`  
Ширина угла в каждую сторону
- `float const VIEW_ANGLE_SHIFT { 0.f }`  
Сдвиг области обзора

Область видимости при повороте

- `float const ROTATION_VIEW_RADIUS { 50.f }`  
Радиус
- `float const ROTATION_VIEW_ANGLE { 30.5f * DEG_RAD }`  
Ширина угла в каждую сторону
- `float const ROTATION_VIEW_ANGLE_SHIFT { 29.5f * DEG_RAD }`  
Сдвиг области обзора

Область видимости незадолго до поворота

- `float const TURN_VIEW_RADIUS { 75.f }`  
Радиус
- `float const TURN_VIEW_ANGLE { 30.f * DEG_RAD }`  
Ширина угла в каждую сторону
- `float const TURN_VIEW_ANGLE_SHIFT { 10.f * DEG_RAD }`  
Сдвиг области обзора

## Область видимости на нерегулируемом перекрестке

- float const `CROSSROAD_VIEW_RADIUS` { 65.f }  
Радиус
- float const `CROSSROAD_VIEW_ANGLE` { 22.5f \* DEG\_RAD }  
Ширина угла в каждую сторону
- float const `CROSSROAD_VIEW_ANGLE_SHIFT` { -52.5f \* DEG\_RAD }  
Сдвиг области обзора

## Область видимости до перестроения

- float const `LINE_CHANGING_VIEW_RADIUS` { 60.f }  
Радиус
- float const `LINE_CHANGING_VIEW_ANGLE` { 30.f \* DEG\_RAD }  
Ширина угла в каждую сторону
- float const `LINE_CHANGING_VIEW_ANGLE_SHIFT` { 20.f \* DEG\_RAD }  
Сдвиг области обзора

## Значения по умолчанию

- `DirectionSignals` const `DEFAULT DIRECTIONS SIGNALS` = { `NotWorking`, `NotWorking`, `NotWorking` }  
Значения по умолчанию для массива сигналов в одном направлении (светофора в этом направлении нет)
- `CrossroadSignals` const `DEFAULT CROSSROAD SIGNALS`  
Значения по умолчанию для массива сигналов всего перекрестка (светофора на перекрестке нет)
- `SignalSprites` const `DEFAULT SIGNAL SPRITES` = { nullptr, nullptr, nullptr, nullptr, nullptr }  
Пустой массив текстур сигналов для одного типа сигнала одного направления
- `SignalsSprites` const `DEFAULT SIGNALS SPRITES` = { `DEFAULT SIGNAL SPRITES`, `DEFAULT SIGNAL SPRITES`, `DEFAULT SIGNAL SPRITES` }  
Пустой массив текстур сигналов для одного направления
- `DirectionsSignalSprites` const `DEFAULT DIRECTIONS SIGNAL SPRITES`  
Пустой массив текстур сигналов для всего перекрестка

## Маски названий файлов

- std::string const `BACKGROUND_FILENAME_MASK` { "res/background/BackgroundNo%No%.png" }  
Маска файлов фонов
- std::string const `MAP_FILENAME_MASK` { "res/map/MapNo%No%.rtmm" }  
Маска файлов карт
- std::string const `ROAD_FILENAME_MASK` { "res/coating/RoadNo%No%.png" }  
Маска файлов текстур дорог
- std::string const `SIGNAL_FILENAME_MASK` { "res/signal/SignalNo%No%.png" }  
Маска файлов текстур сигналов
- std::string const `BUILDING_FILENAME_MASK` { "res/static/BuildingNo%No%.png" }  
Маска файлов текстур зданий
- std::string const `CAR_FILENAME_MASK` { "res/vehicle/CarNo%No%.png" }  
Маска файлов текстур машин

## Параметры дорог

- std::array< float, 2 > const `ROADS_RESISTANCES`
- std::array< `Directions`, 19 > const `ROADS DIRECTIONS`

## Параметры машин

- std::array< float, 6 > const `CARS_MAX_SPEEDS`
- std::array< float, 6 > const `CARS_ACCELERATIONS`

### 6.1.1 Подробное описание

Пространство имен для проекта RTM.

### 6.1.2 Типы

#### 6.1.2.1 Directions

```
using rtm::Directions = typedef std::array<bool, 8>
```

Массив возможных направлений движений по кусочку объекта

См. также

[AngleType](#)

#### 6.1.2.2 LinesCounts

```
using rtm::LinesCounts = typedef std::array<size_t, 4>
```

Массив количества полос в каждом направлении для перекрестков

См. также

[DirectionType](#)

#### 6.1.2.3 DirectionSignals

```
using rtm::DirectionSignals = typedef std::array<SignalType, 4>
```

Массив сигналов из одного направления в каждое

См. также

[DirectionType](#)

#### 6.1.2.4 CrossroadSignals

```
using rtm::CrossroadSignals = typedef std::array<DirectionSignals, 4>
```

Массив сигналов для всех направлений перекрестка

См. также

[DirectionType](#)

#### 6.1.2.5 SignalSprites

```
using rtm::SignalSprites = typedef std::array<cocos2d::Sprite*, 5>
```

Массив всех текстур сигналов из одного направления в одно

См. также

[SignalType](#)

#### 6.1.2.6 SignalsSprites

```
using rtm::SignalsSprites = typedef std::array<SignalSprites, 3>
```

Массив всех текстур сигналов из одного направления в каждое (вперед, влево, вправо)

См. также

[DirectionSignalIndex](#)

#### 6.1.2.7 DirectionsSignalSprites

```
using rtm::DirectionsSignalSprites = typedef std::array<SignalsSprites, 4>
```

Массив всех текстур сигналов для перекрестка

См. также

[DirectionType](#)

### 6.1.3 Перечисления

#### 6.1.3.1 AngleType

```
enum rtm::AngleType
```

Тип для определения положения некоторых объектов и индикации разрешенных направлений на кусочке объекта



Элементы перечислений

NullAngle	Неинициализированный угол
Up	Вверх
Right	Вправо
Down	Вниз
Left	Влево
UpRight	По диагонали вверх вправо
DownRight	По диагонали вниз влево
DownLeft	По диагонали вниз влево
UpLeft	По диагонали вверх вправо

### 6.1.3.2 DirectionType

enum [rtm::DirectionType](#)

Тип для задания направления движения транспорта

Элементы перечислений

NullDirection	Неинициализированное направление
Upward	Направление вверх
Rightward	Направление вправо
Downward	Направление вниз
Leftward	Направление влево

### 6.1.3.3 CoatingUnionType

enum [rtm::CoatingUnionType](#)

Возможные типы дорожных объединений

Элементы перечислений

NoCoatingUnion	Неинициализированный тип
DrivewayType	Прямая дорога
CrossroadType	Обычный перекресток
TCrossroadType	Т-образный перекресток
TurnType	Поворот

## 6.1.3.4 DirectionSignalIndex

```
enum rtm::DirectionSignalIndex
```

Индексы массива для каждого типа сигнала

Элементы перечислений

ForwardSignalIndex	Сигнал в прямом направлении
LeftwardSignalIndex	Сигнал в при повороте налево
RightwardSignalIndex	Сигнал в при повороте направо

## 6.1.3.5 SignalType

```
enum rtm::SignalType
```

Возможные сигналы светофора

Элементы перечислений

NotWorking	Светофор не работает (равносильно его отсутствию)
Allowed	Зеленый сигнал
Warning	Желтый сигнал
Forbidden	Красный сигнал
Closed	В данном направлении движение запрещено

## 6.1.3.6 StateType

```
enum rtm::StateType
```

Возможные состояния для манёвров (движение, поворот, перестроение)

Элементы перечислений

NotStarted	Не начато (не выполняется)
MustStart	Необходимо начать
Started	Начато
MustStop	Необходимо закончить

## 6.1.3.7 CoatingType

enum [rtm::CoatingType](#)

Индексы, начиная с которых начинаются текстуры покрытий определенного типа

Элементы перечислений

Asphalt Coating	Асфальтовое объект
-----------------	--------------------

## 6.1.3.8 SignalFileId

enum [rtm::SignalFileId](#)

Индексы, начиная с которых начинаются текстуры сигнала определенного типа

Элементы перечислений

ForwardSignalId	Индекс сигнала для движения вперед
LeftwardSignalId	Индекс сигнал для поворота налево
RightwardSignalId	Индекс сигнал для поворота направо

## 6.1.4 Функции

## 6.1.4.1 CheckCollisions()

```
void rtm::CheckCollisions (  
    WorldController *const world )
```

Функция для вычисления столкновений в мире

Аргументы

world	контроллер мира, в котором будут происходить вычисления
-------	---

## 6.1.4.2 SameCoordinates()

```
bool rtm::SameCoordinates (  
    float a,
```

```
float b,
float delta = COORD_DELTA )
```

Функция для сравнения двух координат с определенной точностью

Аргументы

a,b	координаты, которые будут сравниваться
delta	максимальная разность между координатами

Возвращает

результат сравнения

#### 6.1.4.3 RoundCoordinate()

```
float rtm::RoundCoordinate (
    float coordinate,
    float delta = COORD_DELTA )
```

Функция пытается округлить координаты до центра клетки

Аргументы

coordinate	координата, которую будем пытаться округлить
delta	максимальное расстояние до центра клетки

Возвращает

если координата достаточно близка к центру, то координаты центра, иначе саму координату

#### 6.1.4.4 RoundToCenter()

```
float rtm::RoundToCenter (
    float coordinate )
```

Функция для округления координаты до центра клетки

Аргументы

coordinate	округляемая координата
------------	------------------------

Возвращает

координата ближайшего центра клетки

#### 6.1.4.5 InCenter()

```
bool rtm::InCenter (
    float coordinate,
    float delta = COORD_DELTA )
```

Функция для проверки координаты на центральность

Аргументы

coordinate	координата, которую проверяем
delta	максимальное расстояние до центра клетки

Возвращает

true, если в центре клетки, иначе false

#### 6.1.4.6 DistanceToNextCenter()

```
float rtm::DistanceToNextCenter (
    float x,
    float y,
    float angle )
```

Функция для нахождения расстояния до следующего центра клетка по ходу движения

Аргументы

x,y	координаты объекта
angle	направление движения (угол)

Возвращает

расстояние до центра

#### 6.1.4.7 CenterIsCrossed()

```
bool rtm::CenterIsCrossed (
    float x,
```

```
float y,
float angle,
float lastDelta )
```

Функция проверяет, пересек ли объект центр клетки (центральную линию, перпендикулярную направлению движения)

Аргументы

x,y	координаты объекта
angle	направление движения (угол)
lastDelta	расстояние, которое объект прошёл за последнее перемещение

Возвращает

true, если пересек какой-либо центр

#### 6.1.4.8 SameAngles()

```
bool rtm::SameAngles (
    float a,
    float b,
    float delta = ANGLE\_DELTA )
```

Функция для сравнения двух углов

Аргументы

a,b	углы, которые будут сравниваться
delta	максимальная разность между углами

Возвращает

результат сравнения

#### 6.1.4.9 RoundAngle()

```
float rtm::RoundAngle (
    float angle,
    float delta = ANGLE\_DELTA )
```

Функция пытается округлить угол до одного из главных направлений (период  $\pi/4$ , т.е. 0,  $\pi/4$ ,  $\pi/2$ , ...)

Аргументы

angle	угол, который будем пытаться округлить
delta	максимальная разность между исходным углом и округленным углом

Возвращает

округленный угол, если исходный был достаточно близок, иначе исходный угол

#### 6.1.4.10 NormalizeAngle()

```
float rtm::NormalizeAngle (  
    float angle )
```

Функция для нормализации угла до диапазона  $[-\pi/2; \pi/2)$

Аргументы

angle	угол, который будем нормализовывать
-------	-------------------------------------

Возвращает

нормализованный угол

#### 6.1.4.11 PixelToCell()

```
int rtm::PixelToCell (  
    float coordinate )
```

Функция для конвертации координаты в номер ячейки

Аргументы

coordinate	координата, которая будет конвертирована
------------	--

Возвращает

номер ячейки

## 6.1.4.12 CellToPixel()

```
float rtm::CellToPixel (  
    int cellNumber )
```

Функция для конвертации номера ячейки в координату центра

Аргументы

cellNumber	номер ячейки, который будет конвертирован
------------	---

Возвращает

координата центра

## 6.1.4.13 AngleToAngleType()

```
rtm::AngleType rtm::AngleToAngleType (  
    float angle )
```

Функция для конвертации угла в угловой тип

Аргументы

angle	угол, который будет конвертирован
-------	-----------------------------------

Возвращает

соответствующий угловой тип

## 6.1.4.14 AngleToDirection()

```
rtm::DirectionType rtm::AngleToDirection (  
    float angle )
```

Функция для конвертации угла в направление

Аргументы

angle	угол, который будет конвертирован
-------	-----------------------------------



Возвращает

соответствующее направление

#### 6.1.4.15 AngleTypeToAngle()

```
float rtm::AngleTypeToAngle (  
    AngleType angle )
```

Функция для конвертации углового типа в угол

Аргументы

angle	угловой тип, который будет конвертирован
-------	--

Возвращает

соответствующий угол

#### 6.1.4.16 AngleTypeToDirection()

```
rtm::DirectionType rtm::AngleTypeToDirection (  
    AngleType angle )
```

Функция для конвертации углового типа в направление

Аргументы

angle	угловой тип, который будет конвертирован
-------	--

Возвращает

соответствующее направление

#### 6.1.4.17 DirectionToAngle()

```
float rtm::DirectionToAngle (  
    DirectionType direction )
```

Функция для конвертации направления в угол

## Аргументы

direction	направление, которое будет конвертировано
-----------	---

## Возвращает

соответствующий угол

## 6.1.4.18 DirectionToAngleType()

```
rtm::AngleType rtm::DirectionToAngleType (  
    DirectionType direction )
```

Функция для конвертации направления в угловой тип

## Аргументы

direction	направление, которое будет конвертировано
-----------	---

## Возвращает

соответствующий угловой тип

## 6.1.4.19 SumAngleTypes()

```
rtm::AngleType rtm::SumAngleTypes (  
    AngleType a,  
    AngleType b )
```

Функция для суммирования двух угловых типов

## Аргументы

a,b	угловые типы, которые будут складываться
-----	--

## Возвращает

сумма угловых типов (a + b)

## 6.1.4.20 CountDeceleration()

```
float rtm::CountDeceleration (
    float maxSpeed )
```

Функция для подсчёта рекомендуемого коэффициента замедления транспорта

Аргументы

maxSpeed	максимальная скорость транспорта
----------	----------------------------------

Возвращает

рекомендуемый коэффициент замедления

## 6.1.5 Переменные

## 6.1.5.1 NEAR\_DELTA

```
float const rtm::NEAR_DELTA { 1.f }
```

Максимальное расстояние до объектов, которые недалеко

См. также

[DynamicObject](#)

## 6.1.5.2 DEFAULT\_CROSSROAD\_SIGNALS

```
CrossroadSignals const rtm::DEFAULT_CROSSROAD_SIGNALS
```

Инициализатор

```
= {
    DEFAULT\_DIRECTIONS\_SIGNALS
    , DEFAULT\_DIRECTIONS\_SIGNALS
    , DEFAULT\_DIRECTIONS\_SIGNALS
    , DEFAULT\_DIRECTIONS\_SIGNALS
}
```

Значения по умолчанию для массива сигналов всего перекрестка (светофора на перекрестке нет)

### 6.1.5.3 DEFAULT\_DIRECTIONS\_SIGNAL\_SPRITES

`DirectionsSignalSprites` const rtm::DEFAULT\_DIRECTIONS\_SIGNAL\_SPRITES

Инициализатор

```
= {
    DEFAULT_SIGNALS_SPRITES
  , DEFAULT_SIGNALS_SPRITES
  , DEFAULT_SIGNALS_SPRITES
  , DEFAULT_SIGNALS_SPRITES
}
```

Пустой массив текстур сигналов для всего перекрестка

### 6.1.5.4 ROADS\_RESISTANCES

`std::array<float, 2>` const rtm::ROADS\_RESISTANCES

Инициализатор

```
= {
    0.f
  , 1.f
}
```

Массив коэффициентов трения для каждого типа объекта

См. также

[CoatingType](#)

### 6.1.5.5 ROADS DIRECTIONS

`std::array<Directions, 19>` const rtm::ROADS DIRECTIONS

Инициализатор

```
= {
    Directions{ false, false, false, false, false, false, false, false }
  , Directions{ true, false, true, false, false, false, false, false }
  , Directions{ true, false, true, false, true, true, false, false }
  , Directions{ true, false, true, false, true, true, true, true }
  , Directions{ true, true, true, true, false, false, false, false }
  , Directions{ true, true, true, true, false, false, false, false }
  , Directions{ true, true, true, true, false, false, false, false }
  , Directions{ true, true, true, true, false, false, false, false }
  , Directions{ false, true, true, true, false, false, false, false }
  , Directions{ false, true, true, true, false, false, false, false }
  , Directions{ false, true, true, true, false, false, false, false }
  , Directions{ false, true, true, true, false, false, false, false }
  , Directions{ true, false, false, false, false, false, true, true }
  , Directions{ false, false, true, false, false, false, true, true }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, true, true, false, false, false, false, false }
  , Directions{ false, false, false, false, false, false, false, false }
}
```

Массив возможных направлений для каждой типа кучоска дороги

См. также

[RoadCoating](#)

#### 6.1.5.6 CARS\_MAX\_SPEEDS

```
std::array<float, 6> const rtm::CARS_MAX_SPEEDS
```

Инициализатор

```
= {  
    0.f  
    , 21.f  
    , 24.f  
    , 30.f  
    , 33.f  
    , 36.f  
}
```

Массив максимальных скоростей для машин

См. также

[CarObject](#)

#### 6.1.5.7 CARS\_ACCELERATIONS

```
std::array<float, 6> const rtm::CARS_ACCELERATIONS
```

Инициализатор

```
= {  
    0.f  
    , 3.f  
    , 4.f  
    , 6.f  
    , 8.25f  
    , 12.f  
}
```

Массив ускорений для машин

См. также

[CarObject](#)



## Глава 7

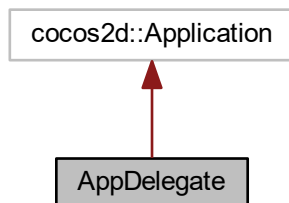
# Классы

### 7.1 Класс AppDelegate

Приложение, основанное на Cocos2d.

```
#include <AppDelegate.h>
```

Граф наследования: AppDelegate:



Открытые члены

- [AppDelegate \(\)](#)  
Конструктор по умолчанию
- [virtual ~AppDelegate \(\)](#)  
Деструктор
- [virtual void initWithGLContextAttrs \(\)](#)  
Функция для установки атрибутов OpenGL (красный, зеленый, синий, альфа-канал...)
- [virtual bool applicationDidFinishLaunching \(\)](#)
- [virtual void applicationDidEnterBackground \(\)](#)  
Функция вызывается, когда приложение скрывается
- [virtual void applicationWillEnterForeground \(\)](#)  
Функция вызывается при первом запуске приложения

### 7.1.1 Подробное описание

Приложение, основанное на Cocos2d.

### 7.1.2 Методы

#### 7.1.2.1 applicationDidFinishLaunching()

```
bool AppDelegate::applicationDidFinishLaunching ( ) [virtual]
```

Функция для инициализации Director'a и Scene'ы

Возвращает

true Инициализация успешна, приложение продолжает выполняться  
false Инициализация провалилась, приложение закроется

Объявления и описания членов классов находятся в файлах:

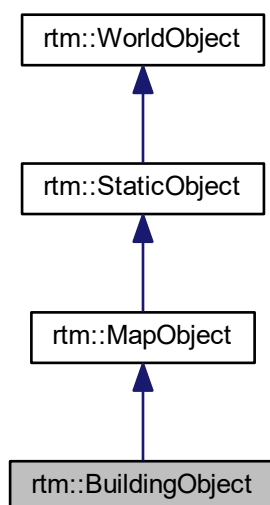
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/AppDelegate.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/AppDelegate.cpp

## 7.2 Класс rtm::BuildingObject

Класс, описывающий строения (здания)

```
#include <BuildingObject.h>
```

Граф наследования:rtm::BuildingObject:





## Открытые члены

- [BuildingObject](#) ()  
Конструктор по умолчанию
- [BuildingObject](#) (cocos2d::Sprite \*const sprite, int column, int row, float angle)
- [BuildingObject](#) (std::string const &filename, int column, int row, float angle)
- [BuildingObject](#) (size\_t type, int column, int row, float angle)
- virtual [~BuildingObject](#) ()=default  
Деструктор по умолчанию

## Закрытые статические члены

- static std::string [GetClassFile\\_](#) (size\_t id)

## Дополнительные унаследованные члены

## 7.2.1 Подробное описание

Класс, описывающий строения (здания)

## 7.2.2 Конструктор(ы)

## 7.2.2.1 BuildingObject() [1/3]

```
rtm::BuildingObject::BuildingObject (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    float angle )
```

Конструктор с использованием уже готового спрайта

## Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать строение
row	строка, в которой необходимо отрисовать строение
angle	угол поворота строения

## 7.2.2.2 BuildingObject() [2/3]

```
rtm::BuildingObject::BuildingObject (
    std::string const & filename,
```

```

    int column,
    int row,
    float angle )

```

### Конструктор из файла

#### Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать строение
row	строка, в которой необходимо отрисовать строение
angle	угол поворота строения

#### 7.2.2.3 BuildingObject() [3/3]

```

rtm::BuildingObject::BuildingObject (
    size_t type,
    int column,
    int row,
    float angle )

```

### Конструктор стандартного строения

#### Аргументы

type	стандартный тип строения
column	колонка, в которой необходимо отрисовать строение
row	строка, в которой необходимо отрисовать строение
angle	угол поворота строения

## 7.2.3 Методы

#### 7.2.3.1 GetClassFile\_()

```

std::string rtm::BuildingObject::GetClassFile_ (
    size_t id ) [static], [private]

```

Функция для получения пути к файлу стандартного строения по номеру

#### Аргументы

id	номер стандартного строения
----	-----------------------------

Возвращает

путь к файлу строения

Объявления и описания членов классов находятся в файлах:

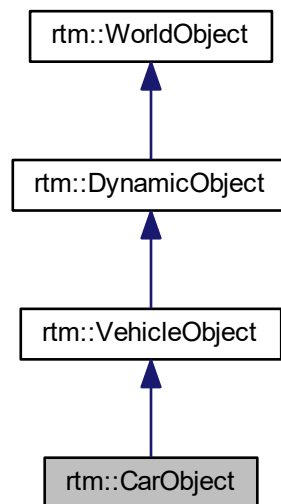
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BuildingObject.h`
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BuildingObject.cpp`

## 7.3 Класс `rtm::CarObject`

Класс, описывающий машины

```
#include <CarObject.h>
```

Граф наследования: `rtm::CarObject`:



Открытые члены

- `CarObject ()`  
Конструктор по умолчанию
- `CarObject (cocos2d::Sprite *const sprite, int column, int row, float angle, float maxSpeed, float acceleration)`
- `CarObject (std::string const &filename, int column, int row, float angle, float maxSpeed, float acceleration)`
- `CarObject (size_t type, int column, int row, float angle)`
- `virtual ~CarObject ()=default`  
Деструктор по умолчанию

### Защищенные члены

- virtual bool `MovementStart_` (`WorldController *const world`) override
- virtual bool `MovementTick_` (`WorldController *const world`) override
- virtual bool `MovementEnd_` (`WorldController *const world`) override
- virtual bool `LineChangingStart` (`WorldController *const world`) override

### Закрытые члены

- void `SetDesiredSpeed_` (float speed)  
Функция для установки желаемой скорости
- void `ResetDesiredSpeed_` ()  
Функция для сброса желаемой скорости
- void `CheckCoatingAhead_` (`WorldController *const world`)
- void `CheckCoatingUnionAhead_` (`WorldController *const world`)
- void `CheckRoadAhead_` (`WorldController *const world`)

### Закрытые статические члены

- static std::string `GetClassFile_` (size\_t id)
- static float `GetClassMaxSpeed_` (size\_t id)
- static float `GetClassAcceleration_` (size\_t id)

### Закрытые данные

- float `recommendedSpeed_`  
Рекомендованная скорость
- float `desiredSpeed_`  
Желаемая скорость (приоритетнее рекомендуемой)
- bool `hasDesiredSpeed_`  
Задана ли желаемая скорость
- bool `isTurnNear_`  
Далеко ли следующий поворот
- bool `isRightTurn_`  
Направление следующего поворота
- bool `waitForSignal_`  
Происходит ли сейчас ожидание сигнала светофора
- bool `waitForTurn_`  
Происходит ли сейчас ожидание освобождения нерегулируемого перекрестка
- `AngleType` `desiredDirection_`  
Желаемое направление движения (при первой возможности машина повернет)

## 7.3.1 Подробное описание

Класс, описывающий машины

## 7.3.2 Конструктор(ы)

## 7.3.2.1 CarObject() [1/3]

```
rtm::CarObject::CarObject (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    float angle,
    float maxSpeed,
    float acceleration )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать машину
row	строка, в которой необходимо отрисовать машину
angle	угол поворота машины
maxSpeed	максимальная скорость машины
acceleration	ускорение машины

## 7.3.2.2 CarObject() [2/3]

```
rtm::CarObject::CarObject (
    std::string const & filename,
    int column,
    int row,
    float angle,
    float maxSpeed,
    float acceleration )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать машину
row	строка, в которой необходимо отрисовать машину
angle	угол поворота машины
maxSpeed	максимальная скорость машины
acceleration	ускорение машины

## 7.3.2.3 CarObject() [3/3]

```
rtm::CarObject::CarObject (
    size_t type,
```

```
int column,
int row,
float angle )
```

Конструктор стандартной машины

Аргументы

type	стандартный тип машины
column	колонка, в которой необходимо отрисовать машину
row	строка, в которой необходимо отрисовать машину
angle	угол поворота машины

### 7.3.3 Методы

#### 7.3.3.1 MovementStart\_()

```
bool rtm::CarObject::MovementStart_ (
    WorldController *const world ) [override], [protected], [virtual]
```

Функция, которая просто пропускает выполнение родителя

Аргументы

world	контроллер мира, в котором находится объект
-------	---

Переопределяет метод предка [rtm::VehicleObject](#).

#### 7.3.3.2 MovementTick\_()

```
bool rtm::CarObject::MovementTick_ (
    WorldController *const world ) [override], [protected], [virtual]
```

Функция для вычисления скорости

Аргументы

world	контроллер мира, в котором находится объект
-------	---

Переопределяет метод предка [rtm::VehicleObject](#).

## 7.3.3.3 MovementEnd\_()

```
bool rtm::CarObject::MovementEnd_ (
    WorldController *const world ) [override], [protected], [virtual]
```

Функция обнуляет финальную скорость

Аргументы

world	контроллер мира, в котором находится объект
-------	---

Переопределяет метод предка [rtm::VehicleObject](#).

## 7.3.3.4 LineChangingStart()

```
bool rtm::CarObject::LineChangingStart (
    WorldController *const world ) [override], [protected], [virtual]
```

Функция, описывающая движение перед перестроением

Аргументы

world	контроллер мира, в котором находится объект
-------	---

Переопределяет метод предка [rtm::VehicleObject](#).

## 7.3.3.5 CheckCoatingAhead\_()

```
void rtm::CarObject::CheckCoatingAhead_ (
    WorldController *const world ) [private]
```

Функция для проверки объекта (повороты и т.д.)

Аргументы

world	контроллер мира, в котором находится объект
-------	---

## 7.3.3.6 CheckCoatingUnionAhead\_()

```
void rtm::CarObject::CheckCoatingUnionAhead_ (
    WorldController *const world ) [private]
```

Функция для проверки объединения покрытий (заранее тормозим перед светофорами и т.д.)

## Аргументы

world	контроллер мира, в котором находится объект
-------	---

## 7.3.3.7 CheckRoadAhead\_()

```
void rtm::CarObject::CheckRoadAhead_ (
    WorldController *const world ) [private]
```

Функция для проверки дороги спереди (принятие решений)

## Аргументы

world	контроллер мира, в котором находится объект
-------	---

## 7.3.3.8 GetClassFile\_()

```
std::string rtm::CarObject::GetClassFile_ (
    size_t id ) [static], [private]
```

Функция для получения пути к файлу стандартной машины по номеру

## Аргументы

id	номер стандартной машины
----	--------------------------

## Возвращает

путь к файлу машины

## 7.3.3.9 GetClassMaxSpeed\_()

```
float rtm::CarObject::GetClassMaxSpeed_ (
    size_t id ) [static], [private]
```

Функция для получения максимальной скорости стандартной машины по номеру

## Аргументы

id	номер стандартной машины
----	--------------------------



Возвращает

максимальная скорость машины

#### 7.3.3.10 GetClassAcceleration\_()

```
float rtm::CarObject::GetClassAcceleration_(  
    size_t id ) [static], [private]
```

Функция для получения ускорения стандартной машины по номеру

Аргументы

id	номер стандартной машины
----	--------------------------

Возвращает

ускорение машины

Объявления и описания членов классов находятся в файлах:

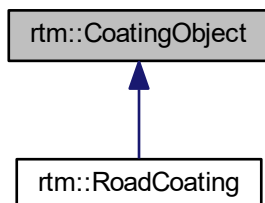
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CarObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CarObject.cpp

## 7.4 Класс rtm::CoatingObject

Класс покрытия секции карты

```
#include <CoatingObject.h>
```

Граф наследования: rtm::CoatingObject:



## Открытые члены

- [CoatingObject](#) ()  
Конструктор по умолчанию
- [CoatingObject](#) (cocos2d::Sprite \*const sprite, int column, int row, [AngleType](#) angle, float resistance, [Directions](#) directions)
- [CoatingObject](#) (std::string const &filename, int column, int row, [AngleType](#) angle, float resistance, [Directions](#) directions)
- virtual [~CoatingObject](#) ()=default  
Деструктор по умолчанию
- cocos2d::Sprite \* [GetSprite](#) () const
- float [GetResistance](#) () const
- bool [HasDirection](#) ([AngleType](#) angle) const
- bool [IsDirectionAvailable](#) ([AngleType](#) angle) const
- void [SetDirectionAvailability](#) ([AngleType](#) angle, bool status)

## Защищенные члены

- void [SetSprite\\_](#) (cocos2d::Sprite \*const sprite)

## Закрытые члены

- void [SetX\\_](#) (float x)
- void [SetY\\_](#) (float y)

## Закрытые данные

- cocos2d::Sprite \* [sprite\\_](#)  
Указатель на спрайт
- float [x\\_](#)  
Абсцисса
- float [y\\_](#)  
Ордината
- float [resistance\\_](#)  
Сопротивление на покрытии
- [Directions](#) [directions\\_](#)  
Доступные направления
- [Directions](#) [availableDirections\\_](#)  
Разрешенные направления

### 7.4.1 Подробное описание

Класс покрытия секции карты

### 7.4.2 Конструктор(ы)

## 7.4.2.1 CoatingObject() [1/2]

```
rtm::CoatingObject::CoatingObject (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    AngleType angle,
    float resistance,
    Directions directions )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать объект
row	строка, в которой необходимо отрисовать объект
angle	угол поворота объекта
resistance	коэффициент сопротивления на покрытии
directions	доступные направления для движения

## 7.4.2.2 CoatingObject() [2/2]

```
rtm::CoatingObject::CoatingObject (
    std::string const & filename,
    int column,
    int row,
    AngleType angle,
    float resistance,
    Directions directions )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать объект
row	строка, в которой необходимо отрисовать объект
angle	угол поворота объекта
resistance	коэффициент сопротивления на покрытии
directions	доступные направления для движения

## 7.4.3 Методы

#### 7.4.3.1 GetSprite()

```
cocos2d::Sprite * rtm::CoatingObject::GetSprite ( ) const
```

Функция для получения спрайта

Возвращает

указатель на спрайт

#### 7.4.3.2 GetResistance()

```
float rtm::CoatingObject::GetResistance ( ) const
```

Функция для получения коэффициента сопротивления на покрытии

Возвращает

сопротивление

#### 7.4.3.3 HasDirection()

```
bool rtm::CoatingObject::HasDirection (   
    AngleType angle ) const
```

Функция для проверки существования направления

Аргументы

angle	направление
-------	-------------

Возвращает

true, если доступно (существует), иначе false

#### 7.4.3.4 IsDirectionAvailable()

```
bool rtm::CoatingObject::IsDirectionAvailable (   
    AngleType angle ) const
```

Функция для проверки разрешенности направления

Аргументы

angle	направление
-------	-------------

Возвращает

true, если разрешено ехать в данном направлении, иначе false

#### 7.4.3.5 SetDirectionAvailability()

```
void rtm::CoatingObject::SetDirectionAvailability (
    AngleType angle,
    bool status )
```

Функция установки разрешенности направления

Аргументы

angle	направление
status	разрешено ли ехать

#### 7.4.3.6 SetSprite\_()

```
void rtm::CoatingObject::SetSprite_ (
    cocos2d::Sprite *const sprite ) [protected]
```

Функция для установки спрайта

Аргументы

sprite	указатель на спрайт
--------	---------------------

#### 7.4.3.7 SetX\_()

```
void rtm::CoatingObject::SetX_ (
    float x ) [private]
```

Функция для установки абсциссы

Аргументы

x	абсцисса
---	----------

#### 7.4.3.8 SetY\_()

```
void rtm::CoatingObject::SetY_ (
    float y ) [private]
```

Функция для установки ординаты

Аргументы

y	ордината
---	----------

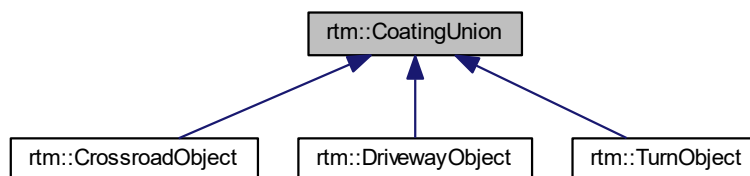
Объявления и описания членов классов находятся в файлах:

- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingObject.cpp

## 7.5 Класс rtm::CoatingUnion

```
#include <CoatingUnion.h>
```

Граф наследования: rtm::CoatingUnion:



Открытые члены

- [CoatingUnion](#) ()  
Конструктор по умолчанию
- [CoatingUnion](#) ([CoatingUnionType](#) type, int column, int row, [CoatingMatrix](#) &&objects)
- [virtual ~CoatingUnion](#) ()=default  
Деструктор по умолчанию

- [CoatingUnionType GetType](#) () const
- [size\\_t GetWidth](#) () const
- [size\\_t GetHeight](#) () const
- [CoatingUnique](#) const & [GetCoatingObject](#) (int column, int row) const
- virtual float [GetLength](#) () const
- bool [IsCorrectColumn](#) (int column) const
- bool [IsCorrectRow](#) (int row) const
- virtual void [ShowSprites](#) (cocos2d::Layer \*const layer)
- virtual void [ReleaseSprites](#) (cocos2d::Layer \*const layer)

### Защищенные члены

- int [GetColumn\\_](#) () const
- int [GetRow\\_](#) () const

### Закрытые данные

- [CoatingUnionType type\\_](#)  
Тип объединения покрытий (получившегося элемента)
- int [column\\_](#)  
Левая колонка объединения
- int [row\\_](#)  
Нижняя строка объединения
- [size\\_t width\\_](#)  
Ширина объединения
- [size\\_t height\\_](#)  
Высота объединения
- [CoatingMatrix objects\\_](#)  
@Матрица объектов покрытий

## 7.5.1 Подробное описание

### Класс объединения покрытий

См. также

[CoatingObject](#)

## 7.5.2 Конструктор(ы)

### 7.5.2.1 CoatingUnion()

```
rtm::CoatingUnion::CoatingUnion (
    CoatingUnionType type,
    int column,
    int row,
    CoatingMatrix && objects )
```

Конструктор по матрице покрытий

## Аргументы

type	тип объединения покрытий (получившегося элемента)
column	левая колонка объединения
row	нижняя строка объединения
objects	матрица объектов покрытий

## 7.5.3 Методы

## 7.5.3.1 GetType()

```
rtm::CoatingUnionType rtm::CoatingUnion::GetType ( ) const
```

Функция для получения типа объединения

Возвращает

тип объединения

## 7.5.3.2 GetWidth()

```
size_t rtm::CoatingUnion::GetWidth ( ) const
```

Функция для получения ширины объединения

Возвращает

ширина объединения

## 7.5.3.3 GetHeight()

```
size_t rtm::CoatingUnion::GetHeight ( ) const
```

Функция для получения высоты объединения

Возвращает

высота объединения

## 7.5.3.4 GetCoatingObject()

```
rtm::CoatingUnique const & rtm::CoatingUnion::GetCoatingObject (
    int column,
    int row ) const
```

Функция для получения объекта



## Аргументы

<code>column</code>	колонка (относительно всей карты), в которой находится объект
<code>row</code>	строка (относительно всей карты), в которой находится объект

Возвращает

умный указатель на объект

7.5.3.5 `GetLength()`

```
float rtm::CoatingUnion::GetLength ( ) const [virtual]
```

Функция для получения длины (количества покрытий)

Возвращает

длина

Переопределяется в [rtm::DrivewayObject](#).

7.5.3.6 `IsCorrectColumn()`

```
bool rtm::CoatingUnion::IsCorrectColumn (
    int column ) const
```

Функция для проверки корректности колонки в данном объединении

Возвращает

`true`, если объединение содержит данную колонку, иначе `false`

7.5.3.7 `IsCorrectRow()`

```
bool rtm::CoatingUnion::IsCorrectRow (
    int row ) const
```

Функция для проверки корректности строки в данном объединении

Возвращает

`true`, если объединение содержит данную строку, иначе `false`

7.5.3.8 `ShowSprites()`

```
void rtm::CoatingUnion::ShowSprites (
    cocos2d::Layer *const layer ) [virtual]
```

Функция для добавления спрайтов на сцену

## Аргументы

layer	слой, на который надо добавить спрайты управляющего блока
-------	---

Переопределяется в [rtm::CrossroadObject](#).

## 7.5.3.9 ReleaseSprites()

```
void rtm::CoatingUnion::ReleaseSprites (
    cocos2d::Layer *const layer ) [virtual]
```

Функция для удаления спрайтов со сцены

## Аргументы

layer	слой, с которого надо удалить спрайты управляющего блока
-------	--

Переопределяется в [rtm::CrossroadObject](#).

## 7.5.3.10 GetColumn\_()

```
int rtm::CoatingUnion::GetColumn_ ( ) const [protected]
```

Функция для получения левой колонки объединения

Возвращает

левая колонка объединения

## 7.5.3.11 GetRow\_()

```
int rtm::CoatingUnion::GetRow_ ( ) const [protected]
```

Функция для получения нижней строки объединения

Возвращает

нижняя строка объединения

Объявления и описания членов классов находятся в файлах:

- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingUnion.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingUnion.cpp

## 7.6 Класс rtm::ControlUnit

Класс управляющего блока (светофор)

```
#include <ControlUnit.h>
```

### Открытые члены

- [ControlUnit](#) ()  
Конструктор по умолчанию
- [ControlUnit](#) (size\_t type, int column, int row, [LinesCounts](#) linesCounts)
- [ControlUnit](#) (size\_t type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection)
- virtual [~ControlUnit](#) ()=default  
Деструктор по умолчанию
- void [Update](#) ([WorldController](#) \*const world)
- [operator bool](#) () const
- [SignalType](#) [GetSignal](#) ([DirectionType](#) from, [DirectionType](#) to) const
- void [ShowSprites](#) (cocos2d::Layer \*const layer)
- void [ReleaseSprites](#) (cocos2d::Layer \*const layer)

### Закрытые члены

- void [InitSignals\\_](#) ()  
Функция для инициализации сигналов
- void [ResetSprites\\_](#) ()  
Функция для отображения спрайтов в зависимости от массива сигналов
- void [UpdateSignal\\_](#) (size\_t i, size\_t j, [SignalType](#) signal)
- void [IncState\\_](#) ()  
Функция для инкремента номера состояния
- void [SetState\\_](#) (size\_t state)
- void [ResetState\\_](#) ()  
Функция для сброса номера состояния

### Закрытые статические члены

- static std::string [GetSignalFile\\_](#) (size\_t id)

### Закрытые данные

- size\_t [type\\_](#)  
Номер типа управляющего блока (задает логику)
- int [column\\_](#)  
Левая колонка перекрестка
- int [row\\_](#)  
Нижняя строка перекрестка
- [LinesCounts](#) [linesCounts\\_](#)  
Количество полос в каждом направлении
- [AngleType](#) [nullDirection\\_](#)  
Сторона, в направлении которой нельзя двигаться

- [CrossroadSignals signals\\_](#)  
Все сигналы управляющего блока
- [DirectionsSignalSprites sprites\\_](#)  
Все спрайты управляющего блока
- float [time\\_](#)  
Время, прошедшее с последней смены сигнала
- size\_t [state\\_](#)  
Номер состояния (для последовательного включения сигналов разных направлений)

### 7.6.1 Подробное описание

Класс управляющего блока (светофор)

### 7.6.2 Конструктор(ы)

#### 7.6.2.1 ControlUnit() [1/2]

```
rtm::ControlUnit::ControlUnit (
    size_t type,
    int column,
    int row,
    LinesCounts linesCounts )
```

Конструктор управляющего блока перекрестком

Аргументы

type	номер типа управляющего блока
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении

#### 7.6.2.2 ControlUnit() [2/2]

```
rtm::ControlUnit::ControlUnit (
    size_t type,
    int column,
    int row,
    LinesCounts linesCounts,
    AngleType nullDirection )
```

Конструктор управляющего блока перекрестком

## Аргументы

type	номер типа управляющего блока
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении
nullDirection	сторона, в направлении которой нельзя двигаться

## 7.6.3 Методы

## 7.6.3.1 Update()

```
void rtm::ControlUnit::Update (  
    WorldController *const world )
```

Функция обновления

## Аргументы

world	контроллер мира, в котором находится объект
-------	---

## 7.6.3.2 operator bool()

```
rtm::ControlUnit::operator bool ( ) const
```

Оператор преобразования в логический тип

Возвращает

true, если управляющий блок работает (меняются сигналы), иначе false

## 7.6.3.3 GetSignal()

```
rtm::SignalType rtm::ControlUnit::GetSignal (  
    DirectionType from,  
    DirectionType to ) const
```

Функция для получения сигнала (статуса направления)

## Аргументы

from	направление, в котором транспорт движется
to	направление, в котором транспорт поедет дальше

## Возвращает

сигнал в нужном направлении

## 7.6.3.4 ShowSprites()

```
void rtm::ControlUnit::ShowSprites (
    cocos2d::Layer *const layer )
```

Функция для добавления спрайтов на сцену

## Аргументы

layer	слой, на который надо добавить спрайты управляющего блока
-------	---

## 7.6.3.5 ReleaseSprites()

```
void rtm::ControlUnit::ReleaseSprites (
    cocos2d::Layer *const layer )
```

Функция для удаления спрайтов со сцены

## Аргументы

layer	слой, с которого надо удалить спрайты управляющего блока
-------	--

## 7.6.3.6 UpdateSignal\_()

```
void rtm::ControlUnit::UpdateSignal_ (
    size_t i,
    size_t j,
    SignalType signal ) [private]
```

Функция для безопасной установки сигнала (если направление не закрыто)

## Аргументы

i	индекс массива для исходного направления
j	индекс массива для конечного направления
signal	новый сигнал

## 7.6.3.7 SetState\_()

```
void rtm::ControlUnit::SetState_ (
    size_t state ) [private]
```

Функция для установки номера состояния

## Аргументы

state	новый номера состояния
-------	------------------------

## 7.6.3.8 GetSignalFile\_()

```
std::string rtm::ControlUnit::GetSignalFile_ (
    size_t id ) [static], [private]
```

Функция для получения пути к файлу сигнала по номеру

## Аргументы

id	номер стандартного сигнала
----	----------------------------

Возвращает

путь к файлу сигнала

Объявления и описания членов классов находятся в файлах:

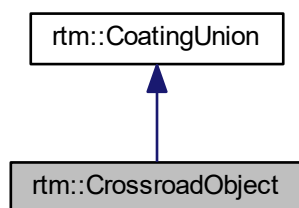
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/ControlUnit.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/ControlUnit.cpp

## 7.7 Класс rtm::CrossroadObject

Класс пересечения дорог

```
#include <CrossroadObject.h>
```

Граф наследования: rtm::CrossroadObject:



### Открытые члены

- [CrossroadObject](#) ()  
Конструктор по умолчанию
- [CrossroadObject](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, size\_t controlUnitType=0)
- [CrossroadObject](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection, size\_t controlUnitType=0)
- virtual [~CrossroadObject](#) ()=default  
Деструктор по умолчанию
- [AngleType](#) [GetNullDirection](#) () const
- [ControlUnitShared](#) [GetControlUnit](#) () const
- virtual void [ShowSprites](#) (cocos2d::Layer \*const layer) override
- virtual void [ReleaseSprites](#) (cocos2d::Layer \*const layer) override

### Открытые статические члены

- static [CoatingMatrix](#) [CrossroadMatrix](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts)
- static [CoatingMatrix](#) [TCrossroadMatrix](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection)

### Закрытые данные

- [LinesCounts](#) [linesCounts\\_](#)  
Количество полос в каждом направлении
- [AngleType](#) [nullDirection\\_](#)  
Сторона, в направлении которой нельзя двигаться
- [ControlUnitShared](#) [controlUnit\\_](#)  
Умный указатель на управляющий блок

### Дополнительные унаследованные члены

#### 7.7.1 Подробное описание

Класс пересечения дорог



## 7.7.2 Конструктор(ы)

### 7.7.2.1 `CrossroadObject()` [1/2]

```
rtm::CrossroadObject::CrossroadObject (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts,
    size_t controlUnitType = 0 )
```

Конструктор для обычного перекрестка

Аргументы

<code>type</code>	тип покрытия
<code>column</code>	левая колонка перекрестка
<code>row</code>	нижняя строка перекрестка
<code>linesCounts</code>	количество полос в каждом направлении
<code>controlUnitType</code>	номер типа управляющего блока

### 7.7.2.2 `CrossroadObject()` [2/2]

```
rtm::CrossroadObject::CrossroadObject (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts,
    AngleType nullDirection,
    size_t controlUnitType = 0 )
```

Конструктор для Т-образного перекрестка

Аргументы

<code>type</code>	тип покрытия
<code>column</code>	левая колонка перекрестка
<code>row</code>	нижняя строка перекрестка
<code>linesCounts</code>	количество полос в каждом направлении
<code>nullDirection</code>	сторона, в направлении которой нельзя двигаться
<code>controlUnitType</code>	номер типа управляющего блока

## 7.7.3 Методы

## 7.7.3.1 CrossroadMatrix()

```
rtm::CoatingMatrix rtm::CrossroadObject::CrossroadMatrix (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts ) [static]
```

Функция для получения матрицы покрытий перекрестка

Аргументы

type	тип покрытия
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении

Возвращает

матрица покрытий

## 7.7.3.2 TCrossroadMatrix()

```
rtm::CoatingMatrix rtm::CrossroadObject::TCrossroadMatrix (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts,
    AngleType nullDirection ) [static]
```

Функция для получения матрицы покрытий Т-образного перекрестка

Аргументы

type	тип покрытия
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении
nullDirection	сторона, в направлении которой нельзя двигаться

Возвращает

матрица покрытий

7.7.3.3 `GetNullDirection()`

```
rtm::AngleType rtm::CrossroadObject::GetNullDirection ( ) const
```

Функция для получения стороны, в направлении которой нельзя двигаться

Возвращает

угол, соответствующий запрещенной стороне

7.7.3.4 `GetControlUnit()`

```
rtm::ControlUnitShared rtm::CrossroadObject::GetControlUnit ( ) const
```

Функция для получения управляющего блока, привязанного к данному объекту

Возвращает

умный указатель на управляющий блок

7.7.3.5 `ShowSprites()`

```
void rtm::CrossroadObject::ShowSprites (
    cocos2d::Layer *const layer ) [override], [virtual]
```

Функция для добавления спрайтов на сцену

Аргументы

<code>layer</code>	слой, на который надо добавить спрайты управляющего блока
--------------------	---

Переопределяет метод предка [rtm::CoatingUnion](#).

7.7.3.6 `ReleaseSprites()`

```
void rtm::CrossroadObject::ReleaseSprites (
    cocos2d::Layer *const layer ) [override], [virtual]
```

Функция для удаления спрайтов со сцены

## Аргументы

layer	слой, с которого надо удалить спрайты управляющего блока
-------	--

Переопределяет метод предка [rtm::CoatingUnion](#).

Объявления и описания членов классов находятся в файлах:

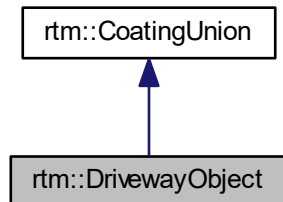
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CrossroadObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CrossroadObject.cpp

## 7.8 Класс rtm::DrivewayObject

Класс прямой дороги

```
#include <DrivewayObject.h>
```

Граф наследования: rtm::DrivewayObject:



### Открытые члены

- [DrivewayObject](#) ()  
Конструктор по умолчанию
- [DrivewayObject](#) ([CoatingType](#) type, int column, int row, size\_t width, size\_t height, [AngleType](#) angle)
- virtual [~DrivewayObject](#) ()=default  
Деструктор по умолчанию
- virtual float [GetLength](#) () const override
- size\_t [GetLinesCount](#) () const
- bool [isRightLine](#) (int column, int row) const
- bool [isRightLine](#) (float x, float y) const
- bool [isLeftLine](#) (int column, int row) const
- bool [isLeftLine](#) (float x, float y) const

## Открытые статические члены

- static [CoatingMatrix DrivewayMatrix](#) ([CoatingType](#) type, int column, int row, size\_t width, size\_t height, [AngleType](#) angle)

## Закрытые статические члены

- static float [CountLength\\_](#) (size\_t width, size\_t height, [AngleType](#) angle)
- static size\_t [CountLines\\_](#) (size\_t width, size\_t height, [AngleType](#) angle)

## Закрытые данные

- [AngleType angle\\_](#)  
Направление движения
- float [length\\_](#)  
Длина объекта (для вычисления кратчайшего пути)
- size\_t [linesCount\\_](#)  
Количество полос

## Дополнительные унаследованные члены

## 7.8.1 Подробное описание

## Класс прямой дороги

## 7.8.2 Конструктор(ы)

## 7.8.2.1 DrivewayObject()

```
rtm::DrivewayObject::DrivewayObject (
    CoatingType type,
    int column,
    int row,
    size_t width,
    size_t height,
    AngleType angle )
```

## Конструктор по размерам

## Аргументы

type	тип покрытия
column	левая колонка объекта
row	нижняя строка объекта
width	ширина объекта
height	высота объекта
angle	направление движения

### 7.8.3 Методы

#### 7.8.3.1 DrivewayMatrix()

```
rtm::CoatingMatrix rtm::DrivewayObject::DrivewayMatrix (
    CoatingType type,
    int column,
    int row,
    size_t width,
    size_t height,
    AngleType angle ) [static]
```

Функция для получения матрицы покрытий дороги

Аргументы

type	тип покрытия
column	левая колонка объекта
row	нижняя строка объекта
width	ширина объекта
height	высота объекта
angle	направление движения

Возвращает

матрица покрытий

#### 7.8.3.2 GetLength()

```
float rtm::DrivewayObject::GetLength ( ) const [override], [virtual]
```

Функция для получения длины объекта (для вычисления кратчайшего пути)

Возвращает

длина

Переопределяет метод предка `rtm::CoatingUnion`.

7.8.3.3 `GetLinesCount()`

```
size_t rtm::DrivewayObject::GetLinesCount ( ) const
```

Функция для получения количества полос

Возвращает

количество полос

7.8.3.4 `isRightLine()` [1/2]

```
bool rtm::DrivewayObject::isRightLine (
    int column,
    int row ) const
```

Функция для проверки: находится ли объект в правой полосе

Аргументы

column	колонка, в которой находится объект
row	строка, в которой находится объект

Возвращает

`true`, если объект находится в правой полосе, иначе `false`

7.8.3.5 `isRightLine()` [2/2]

```
bool rtm::DrivewayObject::isRightLine (
    float x,
    float y ) const
```

Функция для проверки: находится ли объект в правой полосе

Аргументы

x	абсцисса объекта
y	ордината объекта

Возвращает

`true`, если объект находится в правой полосе, иначе `false`

## 7.8.3.6 isLeftLine() [1/2]

```
bool rtm::DrivewayObject::isLeftLine (
    int column,
    int row ) const
```

Функция для проверки: находится ли объект в левой полосе

Аргументы

column	колонка, в которой находится объект
row	строка, в которой находится объект

Возвращает

true, если объект находится в левой полосе, иначе false

## 7.8.3.7 isLeftLine() [2/2]

```
bool rtm::DrivewayObject::isLeftLine (
    float x,
    float y ) const
```

Функция для проверки: находится ли объект в левой полосе

Аргументы

x	абсцисса объекта
y	ордината объекта

Возвращает

true, если объект находится в левой полосе, иначе false

## 7.8.3.8 CountLength\_()

```
float rtm::DrivewayObject::CountLength_ (
    size_t width,
    size_t height,
    AngleType angle ) [static], [private]
```

Функция для вычисления длины объекта



## Аргументы

width	ширина объекта
height	высота объекта
angle	направление движения

## Возвращает

длина объекта

## 7.8.3.9 CountLines\_()

```
size_t rtm::DrivewayObject::CountLines_ (
    size_t width,
    size_t height,
    AngleType angle ) [static], [private]
```

Функция для получения количества полос

## Аргументы

width	ширина объекта
height	высота объекта
angle	направление движения

## Возвращает

количество полос

Объявления и описания членов классов находятся в файлах:

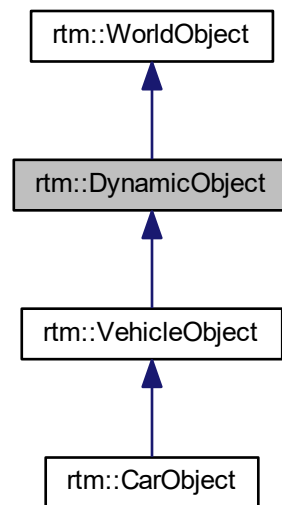
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DrivewayObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DrivewayObject.cpp

## 7.9 Класс rtm::DynamicObject

Класс динамического объекта (который движается, обновляется)

```
#include <DynamicObject.h>
```

Граф наследования: rtm::DynamicObject:



### Открытые члены

- [DynamicObject](#) ()  
Конструктор по умолчанию
- [DynamicObject](#) (cocos2d::Sprite \*sprite, float x, float y, float angle, float speed)
- [DynamicObject](#) (std::string const &filename, float x, float y, float angle, float speed)
- virtual [~DynamicObject](#) ()=default  
Деструктор по умолчанию
- float [GetSpeed](#) () const
- float [GetLastDelta](#) () const
- bool [HasCollision](#) () const
- virtual void [Update](#) ([WorldController](#) \*const world)
- bool [IsNearOthers](#) ([WorldController](#) \*const world)

### Защищенные члены

- void [SetSpeed\\_](#) (float speed)
- void [SetCollisionFlag\\_](#) (bool flag)
- bool [IsBeholding\\_](#) ([WorldObject](#) const \*const other, float radius=[VIEW\\_RADIUS](#), float angle=[VIEW\\_ANGLE](#), float angleShift=[VIEW\\_ANGLE\\_SHIFT](#)) const
- bool [IsIntersecting\\_](#) ([WorldObject](#) const \*const other) const

### Закрытые члены

- bool [IsNear\\_](#) ([WorldObject](#) const \*const other) const

## Закрытые данные

- float [speed\\_](#)  
Текущая скорость
- float [lastDelta\\_](#)  
Длина последнего смещения
- bool [hasCollision\\_](#)  
Наличие столкновений у данного объекта

## Друзья

- void [CheckCollisions](#) ([WorldController](#) \*const world)

### 7.9.1 Подробное описание

Класс динамического объекта (который двигается, обновляется)

### 7.9.2 Конструктор(ы)

#### 7.9.2.1 DynamicObject() [1/2]

```
rtm::DynamicObject::DynamicObject (
    cocos2d::Sprite * sprite,
    float x,
    float y,
    float angle,
    float speed )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
x	абсцисса будущего объекта
y	ордината будущего объекта
angle	угол поворота строения
speed	первоначальная скорость

#### 7.9.2.2 DynamicObject() [2/2]

```
rtm::DynamicObject::DynamicObject (
    std::string const & filename,
```

```
float x,  
float y,  
float angle,  
float speed )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
x	абсцисса будущего объекта
y	ордината будущего объекта
angle	угол поворота строения
speed	первоначальная скорость

### 7.9.3 Методы

#### 7.9.3.1 GetSpeed()

```
float rtm::DynamicObject::GetSpeed ( ) const
```

Функция для получения скорости

Возвращает

скорость объекта

#### 7.9.3.2 GetLastDelta()

```
float rtm::DynamicObject::GetLastDelta ( ) const
```

Функция для получения последнего приращения положения

Возвращает

длина последнего смещения

7.9.3.3 `HasCollision()`

```
bool rtm::DynamicObject::HasCollision ( ) const
```

Функция для проверки наличия столкновений у данного объекта

Возвращает

`true`, если после последней проверки была столкновение, иначе `false`

7.9.3.4 `Update()`

```
void rtm::DynamicObject::Update (
    WorldController *const world ) [virtual]
```

Функция обновления

Аргументы

<code>world</code>	контроллер мира, в котором находится объект
--------------------	---

Переопределяется в [rtm::VehicleObject](#).

7.9.3.5 `IsNearOthers()`

```
bool rtm::DynamicObject::IsNearOthers (
    WorldController *const world )
```

Функция для поиска объектов неподалеку

Аргументы

<code>world</code>	контроллер мира, в котором находится объект
--------------------	---

Возвращает

`true`, если какой-нибудь объект находится рядом, иначе `false`

7.9.3.6 `SetSpeed_()`

```
void rtm::DynamicObject::SetSpeed_ (
    float speed ) [protected]
```

Функция для установки скорости

Аргументы

speed	новая скорость
-------	----------------

### 7.9.3.7 SetCollisionFlag\_()

```
void rtm::DynamicObject::SetCollisionFlag_ (
    bool flag ) [protected]
```

Функция для сохранения информации о столкновениях

Аргументы

flag	есть ли столкновение
------	----------------------

### 7.9.3.8 IsBeholding\_()

```
bool rtm::DynamicObject::IsBeholding_ (
    WorldObject const *const other,
    float radius = VIEW_RADIUS,
    float angle = VIEW_ANGLE,
    float angleShift = VIEW_ANGLE_SHIFT ) const [protected]
```

Функция для проверки попадания объекта в зону видимости

Аргументы

other	указатель на второй объект
radius	радиус видимости
angle	угол видимости (в каждую из сторон)
angleShift	сдвиг области видимости

Возвращает

true, если other находится в области видимости данного объекта, иначе false

### 7.9.3.9 IsIntersecting\_()

```
bool rtm::DynamicObject::IsIntersecting_ (
    WorldObject const *const other ) const [protected]
```

Функция для проверки наличия столкновения с other

## Аргументы

<code>other</code>	указатель на второй объект
--------------------	----------------------------

## Возвращает

`true`, если объект пересекается с `other`, иначе `false`

7.9.3.10 `IsNear_()`

```
bool rtm::DynamicObject::IsNear_ (
    WorldObject const *const other ) const    [private]
```

Функция для проверки, находится ли `other` рядом с данным объектом

## Аргументы

<code>other</code>	указатель на второй объект
--------------------	----------------------------

## Возвращает

`true`, если `other` рядом, иначе `false`

## 7.9.4 Документация по друзьям класса и функциям, относящимся к классу

7.9.4.1 `CheckCollisions`

```
void CheckCollisions (
    WorldController *const world )    [friend]
```

Функция для вычисления столкновений в мире

## Аргументы

<code>world</code>	контроллер мира, в котором будут происходить вычисления
--------------------	---

Объявления и описания членов классов находятся в файлах:

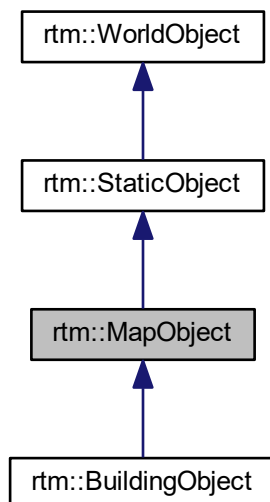
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DynamicObject.h`
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DynamicObject.cpp`

## 7.10 Класс rtm::MapObject

Класс статического объекта карты

```
#include <MapObject.h>
```

Граф наследования: rtm::MapObject:



### Открытые члены

- [MapObject](#) ()  
Конструктор по умолчанию
- [MapObject](#) (cocos2d::Sprite \*sprite, int column, int row, float angle)
- [MapObject](#) (std::string const &filename, int column, int row, float angle)
- `virtual ~MapObject` ()=default  
Деструктор по умолчанию

### Дополнительные унаследованные члены

#### 7.10.1 Подробное описание

Класс статического объекта карты

#### 7.10.2 Конструктор(ы)



## 7.10.2.1 MapObject() [1/2]

```
rtm::MapObject::MapObject (
    cocos2d::Sprite * sprite,
    int column,
    int row,
    float angle )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать объект
row	строка, в которой необходимо отрисовать объект
angle	угол поворота объекта

## 7.10.2.2 MapObject() [2/2]

```
rtm::MapObject::MapObject (
    std::string const & filename,
    int column,
    int row,
    float angle )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать объект
row	строка, в которой необходимо отрисовать объект
angle	угол поворота объекта

Объявления и описания членов классов находятся в файлах:

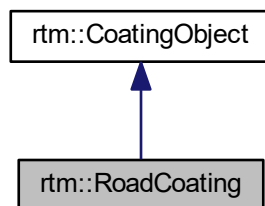
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/MapObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/MapObject.cpp

## 7.11 Класс rtm::RoadCoating

Класс, описывающий дороги

```
#include <RoadCoating.h>
```

Граф наследования:rtm::RoadCoating:



#### Открытые члены

- [RoadCoating](#) ()  
Конструктор по умолчанию
- [RoadCoating](#) (cocos2d::Sprite \*const sprite, int column, int row, [AngleType](#) angle, float resistance, [Directions](#) directions)
- [RoadCoating](#) (std::string const &filename, int column, int row, [AngleType](#) angle, float resistance, [Directions](#) directions)
- [RoadCoating](#) ([CoatingType](#) type, size\_t id, int column, int row, [AngleType](#) angle)
- virtual [~RoadCoating](#) ()=default  
Деструктор по умолчанию

#### Закрытые статические члены

- static std::string [GetClassFile\\_](#) (size\_t id)
- static float [GetClassResistance\\_](#) ([CoatingType](#) type)
- static [Directions](#) const & [GetClassDirections\\_](#) (size\_t id)

#### Дополнительные унаследованные члены

##### 7.11.1 Подробное описание

Класс, описывающий дороги

##### 7.11.2 Конструктор(ы)

###### 7.11.2.1 RoadCoating() [1/3]

```

rtm::RoadCoating::RoadCoating (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    AngleType angle,
    float resistance,
    Directions directions )
  
```

Конструктор с использованием уже готового спрайта

## Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать дорогу
row	строка, в которой необходимо отрисовать дорогу
angle	угол поворота дороги
resistance	коэффициент сопротивления на дороге
directions	доступные направления для движения

## 7.11.2.2 RoadCoating() [2/3]

```
rtm::RoadCoating::RoadCoating (
    std::string const & filename,
    int column,
    int row,
    AngleType angle,
    float resistance,
    Directions directions )
```

## Конструктор из файла

## Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать дорогу
row	строка, в которой необходимо отрисовать дорогу
angle	угол поворота дороги
resistance	коэффициент сопротивления на дороге
directions	доступные направления для движения

## 7.11.2.3 RoadCoating() [3/3]

```
rtm::RoadCoating::RoadCoating (
    CoatingType type,
    size_t id,
    int column,
    int row,
    AngleType angle )
```

## Конструктор стандартного дороги

## Аргументы

type	стандартный тип покрытия
id	номер стандартной дороги

## Аргументы

column	колонка, в которой необходимо отрисовать дорогу
row	строка, в которой необходимо отрисовать дорогу
angle	угол поворота дороги

## 7.11.3 Методы

## 7.11.3.1 GetClassFile\_()

```
std::string rtm::RoadCoating::GetClassFile_(  
    size_t id ) [static], [private]
```

Функция для получения пути к файлу стандартной дороги по номеру

## Аргументы

id	номер стандартной дороги
----	--------------------------

## Возвращает

путь к файлу дороги

## 7.11.3.2 GetClassResistance\_()

```
float rtm::RoadCoating::GetClassResistance_(  
    CoatingType type ) [static], [private]
```

Функция для получения коэффициента сопротивления на стандартной дороге по номеру

## Аргументы

type	тип покрытия
------	--------------

## Возвращает

сопротивление

## 7.11.3.3 GetClassDirections\_()

```
rtm::Directions const & rtm::RoadCoating::GetClassDirections_ (
    size_t id ) [static], [private]
```

Функция для получения доступных направлений стандартной дороги по номеру

Аргументы

id	номер стандартной дороги
----	--------------------------

Возвращает

доступные направления

Объявления и описания членов классов находятся в файлах:

- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/RoadCoating.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/RoadCoating.cpp

## 7.12 Структура rtm::SpawnType

Структура, описывающая параметры точки генерации объектов

```
#include <General.h>
```

Открытые атрибуты

- int **column**  
Номер столбца
- int **row**  
Номер строки
- float **angle**  
Первоначальный угол для транспорта

## 7.12.1 Подробное описание

Структура, описывающая параметры точки генерации объектов

Объявления и описания членов структуры находятся в файле:

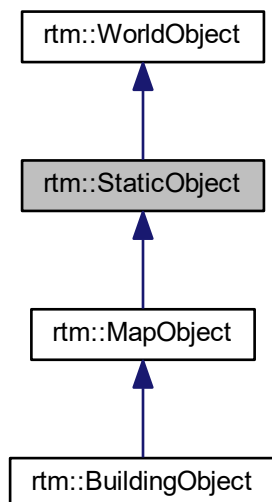
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/General.h

## 7.13 Класс `rtm::StaticObject`

Класс статического объекта (который не обновляется)

```
#include <StaticObject.h>
```

Граф наследования: `rtm::StaticObject`:



### Открытые члены

- [StaticObject](#) ()  
Конструктор по умолчанию
- [StaticObject](#) (`cocos2d::Sprite *sprite`, `float x`, `float y`, `float angle`)
- [StaticObject](#) (`std::string const &filename`, `float x`, `float y`, `float angle`)
- `virtual ~StaticObject` ()=default  
Деструктор по умолчанию

### Дополнительные унаследованные члены

#### 7.13.1 Подробное описание

Класс статического объекта (который не обновляется)

#### 7.13.2 Конструктор(ы)

## 7.13.2.1 StaticObject() [1/2]

```
rtm::StaticObject::StaticObject (
    cocos2d::Sprite * sprite,
    float x,
    float y,
    float angle )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
x	абсцисса будущего объекта
y	ордината будущего объекта
angle	угол поворота строения

## 7.13.2.2 StaticObject() [2/2]

```
rtm::StaticObject::StaticObject (
    std::string const & filename,
    float x,
    float y,
    float angle )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
x	абсцисса будущего объекта
y	ордината будущего объекта
angle	угол поворота строения

Объявления и описания членов классов находятся в файлах:

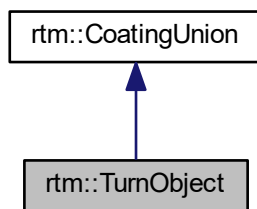
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/StaticObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/StaticObject.cpp

## 7.14 Класс rtm::TurnObject

Класс поворота дороги

```
#include <TurnObject.h>
```

Граф наследования: rtm::TurnObject:



### Открытые члены

- [TurnObject](#) ()  
Конструктор по умолчанию
- [TurnObject](#) (bool isRight, [CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)
- virtual [~TurnObject](#) ()=default  
Деструктор по умолчанию
- bool [IsRight](#) () const
- [AngleType](#) [GetAngle](#) () const

### Открытые статические члены

- static [CoatingMatrix](#) [RightTurnMatrix](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)
- static [CoatingMatrix](#) [LeftTurnMatrix](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)

### Закрытые данные

- bool [isRight\\_](#)  
Тип поворота
- [AngleType](#) [angle\\_](#)  
Угол поворота объекта

### Дополнительные унаследованные члены

#### 7.14.1 Подробное описание

Класс поворота дороги



## 7.14.2 Конструктор(ы)

## 7.14.2.1 TurnObject()

```
rtm::TurnObject::TurnObject (
    bool isRight,
    CoatingType type,
    int column,
    int row,
    size_t linesCount,
    AngleType angle )
```

Конструктор по размерам

Аргументы

isRight	тип поворота (false - левый, true - правый)
type	тип покрытия
column	левая колонка объекта
row	нижняя строка объекта
linesCount	количество полос
angle	угол поворота объекта

## 7.14.3 Методы

## 7.14.3.1 RightTurnMatrix()

```
rtm::CoatingMatrix rtm::TurnObject::RightTurnMatrix (
    CoatingType type,
    int column,
    int row,
    size_t linesCount,
    AngleType angle ) [static]
```

Функция для получения матрицы правого поворота

Аргументы

type	тип покрытия
column	левая колонка объекта
row	нижняя строка объекта
linesCount	количество полос
angle	угол поворота объекта

Возвращает

матрица покрытий

#### 7.14.3.2 LeftTurnMatrix()

```
rtm::CoatingMatrix rtm::TurnObject::LeftTurnMatrix (
    CoatingType type,
    int column,
    int row,
    size_t linesCount,
    AngleType angle ) [static]
```

Функция для получения матрицы левого поворота

Аргументы

type	тип покрытия
column	левая колонка объекта
row	нижняя строка объекта
linesCount	количество полос
angle	угол поворота объекта

Возвращает

матрица покрытий

#### 7.14.3.3 IsRight()

```
bool rtm::TurnObject::IsRight ( ) const
```

Функция для получения типа поворота

Возвращает

true, если правый поворот, false, если левый

#### 7.14.3.4 GetAngle()

```
rtm::AngleType rtm::TurnObject::GetAngle ( ) const
```

Функция для получения угла поворота объекта

Возвращает

угол поворота объекта

Объявления и описания членов классов находятся в файлах:

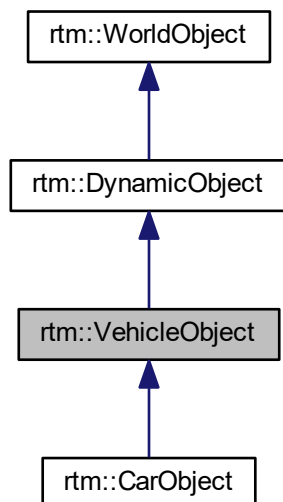
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/TurnObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/TurnObject.cpp

## 7.15 Класс rtm::VehicleObject

Класс транспорта (динамического объекта карты)

```
#include <VehicleObject.h>
```

Граф наследования: rtm::VehicleObject:



### Открытые члены

- [VehicleObject](#) ()  
Конструктор по умолчанию
- [VehicleObject](#) (cocos2d::Sprite \*const sprite, int column, int row, float angle, float maxSpeed, float acceleration, float deceleration)
- [VehicleObject](#) (std::string const &filename, int column, int row, float angle, float maxSpeed, float acceleration, float deceleration)
- virtual [~VehicleObject](#) ()=default  
Деструктор по умолчанию
- virtual void [Update](#) ([WorldController](#) \*const world) override

### Защищенные члены

- bool [MoveForward](#)\_ ()
- bool [Stop](#)\_ ()
- bool [Rotate](#)\_ (float angle=[ANGLE\\_RIGHT](#))
- bool [ChangeLine](#)\_ (bool isRight=[LEFT](#))
- bool [IsMovement](#)\_ () const
- bool [IsRotation](#)\_ () const
- bool [IsLineChanging](#)\_ () const

- bool `IsBraking_` () const
- float `GetMaxSpeed_` () const
- float `GetFinalSpeed_` () const
- void `SetFinalSpeed_` (float speed)
- void `SetBrakingFactor_` (float factor)
- void `StopAtDistance_` (float distance)

### Зрение у машин

Функции для просмотра окружающего мира

- `CoatingObject * CheckForwardCoating_` (`WorldController *const world`, int delta=1)
- `CoatingUnion * CheckForwardCoatingUnion_` (`WorldController *const world`, int delta=1)
- `DynamicObject * CheckForwardArea_` (`WorldController *const world`, float radius, float angle, float angleShift)
- `DynamicObject * CheckMovingArea_` (`WorldController *const world`)
- `DynamicObject * CheckTurnArea_` (`WorldController *const world`, bool isRight)
- `DynamicObject * CheckRotationArea_` (`WorldController *const world`)
- `DynamicObject * CheckCrossroadArea_` (`WorldController *const world`)
- `DynamicObject * CheckLineChangingArea_` (`WorldController *const world`)

### Маневры

Функции выполняющиеся во время маневров

- virtual void `BeforeMoving_` (`WorldController *const world`)
- virtual void `AfterMoving_` (`WorldController *const world`)
- virtual bool `MovementStart_` (`WorldController *const world`)
- virtual bool `MovementTick_` (`WorldController *const world`)
- virtual bool `MovementEnd_` (`WorldController *const world`)
- virtual bool `RotationStart_` (`WorldController *const world`)
- virtual bool `RotationTick_` (`WorldController *const world`)
- virtual bool `RotationEnd_` (`WorldController *const world`)
- virtual bool `LineChangingStart` (`WorldController *const world`)
- virtual bool `LineChangingTick_` (`WorldController *const world`)
- virtual bool `LineChangingEnd_` (`WorldController *const world`)

### Закрытые члены

- void `LineChanging_` (`WorldController *const world`)
- void `Rotation_` (`WorldController *const world`)
- void `Movement_` (`WorldController *const world`)
- void `SpeedChanging_` (`WorldController *const world`)
- void `SmoothBrakingCounter` (`WorldController *const world`)

### Закрытые данные

- `StateType isMovement_`  
Этап выполнения движения (стоит, движется)
- `StateType isRotation_`  
Этап выполнения поворота
- `StateType isLineChanging_`  
Этап выполнения перестроения
- float const `maxSpeed_`  
Максимальная скорость
- float const `acceleration_`

- Ускорение
- float const `deceleration_`  
Скорость замедления
- float `finalSpeed_`  
Финальная скорость
- float `brakingFactor_`  
Коэффициент торможения
- float `brakingDistance_`  
Тормозной путь
- float `rotationAngle_`  
Угол, на который надо повернуться
- float `rotationRadius_`  
Радиус окружности, по которой движется объект
- float `remainingOffset_`  
Оставшийся перпендикулярный движению сдвиг при перестроении
- float `remainingOffsetAngle_`  
Направление перестроения (перпендикулярно движению)

### 7.15.1 Подробное описание

Класс транспорта (динамического объекта карты)

### 7.15.2 Конструктор(ы)

#### 7.15.2.1 VehicleObject() [1/2]

```
rtm::VehicleObject::VehicleObject (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    float angle,
    float maxSpeed,
    float acceleration,
    float deceleration )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
column	колонка, в которой необходимо отрисовать машину
row	строка, в которой необходимо отрисовать машину
angle	угол поворота машины
maxSpeed	максимальная скорость машины
acceleration	ускорение машины
deceleration	скорость замедления машины

### 7.15.2.2 VehicleObject() [2/2]

```
rtm::VehicleObject::VehicleObject (
    std::string const & filename,
    int column,
    int row,
    float angle,
    float maxSpeed,
    float acceleration,
    float deceleration )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
column	колонка, в которой необходимо отрисовать машину
row	строка, в которой необходимо отрисовать машину
angle	угол поворота машины
maxSpeed	максимальная скорость машины
acceleration	ускорение машины
deceleration	скорость замедления машины

## 7.15.3 Методы

### 7.15.3.1 Update()

```
void rtm::VehicleObject::Update (
    WorldController *const world ) [override], [virtual]
```

Функция обновления

Аргументы

world	контроллер мира, в котором находится объект
-------	---

Переопределяет метод предка [rtm::DynamicObject](#).

### 7.15.3.2 MoveForward\_()

```
bool rtm::VehicleObject::MoveForward_ ( ) [protected]
```

Функция для сообщения о необходимости начать движение

Возвращает

`true`, если возможно начать движение, иначе `false`

#### 7.15.3.3 `Stop_()`

`bool rtm::VehicleObject::Stop_ ( ) [protected]`

Функция для сообщения о необходимости остановиться

Возвращает

`true`, если возможно остановиться, иначе `false`

#### 7.15.3.4 `Rotate_()`

`bool rtm::VehicleObject::Rotate_ (`  
    `float angle = ANGLE\_RIGHT ) [protected]`

Функция для сообщения о необходимости повернуть

Возвращает

`true`, если возможно повернуть, иначе `false`

#### 7.15.3.5 `ChangeLine_()`

`bool rtm::VehicleObject::ChangeLine_ (`  
    `bool isRight = LEFT ) [protected]`

Функция для сообщения о необходимости перестроиться

Возвращает

`true`, если возможно перестроиться, иначе `false`

#### 7.15.3.6 IsMovement\_()

```
bool rtm::VehicleObject::IsMovement_ ( ) const [protected]
```

Функция, сообщающая о движении объекта

Возвращает

true, если объект движется, иначе false

#### 7.15.3.7 IsRotation\_()

```
bool rtm::VehicleObject::IsRotation_ ( ) const [protected]
```

Функция, сообщающая о повороте объекта

Возвращает

true, если объект поворачивает, иначе false

#### 7.15.3.8 IsLineChanging\_()

```
bool rtm::VehicleObject::IsLineChanging_ ( ) const [protected]
```

Функция, сообщающая о перестроении объекта

Возвращает

true, если объект перестраивается, иначе false

#### 7.15.3.9 IsBraking\_()

```
bool rtm::VehicleObject::IsBraking_ ( ) const [protected]
```

Функция, сообщающая о торможении объекта перед светофором и т.д.

Возвращает

true, если объект тормозит, иначе false



## 7.15.3.10 GetMaxSpeed\_()

```
float rtm::VehicleObject::GetMaxSpeed_ ( ) const [protected]
```

Функция для получения максимальной скорости

Возвращает

максимальная скорость

## 7.15.3.11 GetFinalSpeed\_()

```
float rtm::VehicleObject::GetFinalSpeed_ ( ) const [protected]
```

Функция для получения финальной скорости (к которой объект будет стремиться)

Возвращает

конечная скорость

## 7.15.3.12 SetFinalSpeed\_()

```
void rtm::VehicleObject::SetFinalSpeed_ (
    float speed ) [protected]
```

Функция для установки финальной скорости (к которой объект будет стремиться)

Аргументы

speed	новая скорость
-------	----------------

## 7.15.3.13 SetBrakingFactor\_()

```
void rtm::VehicleObject::SetBrakingFactor_ (
    float factor ) [protected]
```

Функция для установки коэффициента торможения

Аргументы

factor	новый коэффициент торможения (1 - торможение с обычным ускорением)
--------	--

## 7.15.3.14 StopAtDistance\_()

```
void rtm::VehicleObject::StopAtDistance_ (
    float distance ) [protected]
```

Функция для установки тормозного пути (объект будет пытаться остановиться за данную дистанцию)

Аргументы

distance	дистанция, за которую необходимо остановиться
----------	---

## 7.15.3.15 CheckForwardCoating\_()

```
rtm::CoatingObject * rtm::VehicleObject::CheckForwardCoating_ (
    WorldController *const world,
    int delta = 1 ) [protected]
```

Функция для получения следующего по ходу движения покрытия

Аргументы

world	контроллер мира, в котором находится объект
delta	сдвиг в клетках относительно данного объекта (1 - следующая, 2 - через одну)

Возвращает

указатель на покрытие

## 7.15.3.16 CheckForwardCoatingUnion\_()

```
rtm::CoatingUnion * rtm::VehicleObject::CheckForwardCoatingUnion_ (
    WorldController *const world,
    int delta = 1 ) [protected]
```

Функция для получения следующего по ходу движения объединения покрытий

Аргументы

world	контроллер мира, в котором находится объект
delta	сдвиг в клетках относительно данного объекта (1 - следующее, 2 - через одно)

Возвращает

указатель на объединение покрытий

#### 7.15.3.17 CheckForwardArea\_()

```
rtm::DynamicObject * rtm::VehicleObject::CheckForwardArea_ (
    WorldController *const world,
    float radius,
    float angle,
    float angleShift ) [protected]
```

Функция для проверки области видимости спереди

Аргументы

world	контроллер мира, в котором находится данный объект
radius	радиус видимости
angle	угол видимости (в каждую из сторон)
angleShift	сдвиг области видимости

Возвращает

указатель на объект, находящийся в области видимости  
nullptr, если нет объектов в области видимости

#### 7.15.3.18 CheckMovingArea\_()

```
rtm::DynamicObject * rtm::VehicleObject::CheckMovingArea_ (
    WorldController *const world ) [protected]
```

Функция для проверки области видимости во время движения по прямой

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

указатель на объект, находящийся в области видимости  
nullptr, если нет объектов в области видимости

## 7.15.3.19 CheckTurnArea\_()

```
rtm::DynamicObject * rtm::VehicleObject::CheckTurnArea_ (
    WorldController *const world,
    bool isRight ) [protected]
```

Функция для проверки области видимости перед поворотом

Аргументы

world	контроллер мира, в котором находится данный объект
isRight	сторона, в которую совершается поворот (тип поворота)

Возвращает

указатель на объект, находящийся в области видимости  
 nullptr, если нет объектов в области видимости

## 7.15.3.20 CheckRotationArea\_()

```
rtm::DynamicObject * rtm::VehicleObject::CheckRotationArea_ (
    WorldController *const world ) [protected]
```

Функция для проверки области видимости во время поворота

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

указатель на объект, находящийся в области видимости  
 nullptr, если нет объектов в области видимости

## 7.15.3.21 CheckCrossroadArea\_()

```
rtm::DynamicObject * rtm::VehicleObject::CheckCrossroadArea_ (
    WorldController *const world ) [protected]
```

Функция для проверки области видимости перед нерегулируемым перекрестком

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

указатель на объект, находящийся в области видимости  
`nullptr`, если нет объектов в области видимости

#### 7.15.3.22 `CheckLineChangingArea_()`

```
rtm::DynamicObject * rtm::VehicleObject::CheckLineChangingArea_ (  
    WorldController *const world ) [protected]
```

Функция для проверки области видимости перед перестроением

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

Возвращает

указатель на объект, находящийся в области видимости  
`nullptr`, если нет объектов в области видимости

#### 7.15.3.23 `BeforeMoving_()`

```
void rtm::VehicleObject::BeforeMoving_ (  
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся непосредственно перед перемещением объекта

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

#### 7.15.3.24 `AfterMoving_()`

```
void rtm::VehicleObject::AfterMoving_ (  
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся непосредственно после перемещением объекта

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

## 7.15.3.25 MovementStart\_()

```
bool rtm::VehicleObject::MovementStart_ (
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся перед началом движения

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

true, если шаг успешно завершён (в следующий раз выполнится MovementTick)  
false, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

Переопределяется в `rtm::CarObject`.

## 7.15.3.26 MovementTick\_()

```
bool rtm::VehicleObject::MovementTick_ (
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся во время движения

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

true, если шаг успешно завершён (в следующий раз выполнится MovementEnd)  
false, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

Переопределяется в `rtm::CarObject`.

## 7.15.3.27 MovementEnd\_()

```
bool rtm::VehicleObject::MovementEnd_ (
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся после движения (перед началом остановки)

## Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

## Возвращает

`true`, если шаг успешно завершён (движение на этом закончится)  
`false`, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

Переопределяется в [rtm::CarObject](#).

7.15.3.28 `RotationStart_()`

```
bool rtm::VehicleObject::RotationStart_ (
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся перед поворотом

## Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

## Возвращает

`true`, если шаг успешно завершён (в следующий раз выполнится `RotationTick`)  
`false`, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

7.15.3.29 `RotationTick_()`

```
bool rtm::VehicleObject::RotationTick_ (
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся во время поворота

## Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

## Возвращает

`true`, если шаг успешно завершён (в следующий раз выполнится `RotationEnd`)  
`false`, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

7.15.3.30 `RotationEnd_()`

```
bool rtm::VehicleObject::RotationEnd_ (
    WorldController *const world )    [protected], [virtual]
```

Функция, выполняющаяся после поворота

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

true, если шаг успешно завершён (поворот на этом закончится)  
false, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

7.15.3.31 `LineChangingStart()`

```
bool rtm::VehicleObject::LineChangingStart (
    WorldController *const world )    [protected], [virtual]
```

Функция, выполняющаяся перед перестроением

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Возвращает

true, если шаг успешно завершён (в следующий раз выполнится `LineChangingTick`)  
false, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

Переопределяется в [rtm::CarObject](#).

7.15.3.32 `LineChangingTick_()`

```
bool rtm::VehicleObject::LineChangingTick_ (
    WorldController *const world )    [protected], [virtual]
```

Функция, выполняющаяся во время перестроения

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--



Возвращает

`true`, если шаг успешно завершён (в следующий раз выполнится `LineChangingEnd`)  
`false`, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

#### 7.15.3.33 `LineChangingEnd_()`

```
bool rtm::VehicleObject::LineChangingEnd_ (  
    WorldController *const world ) [protected], [virtual]
```

Функция, выполняющаяся после перестроения

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

Возвращает

`true`, если шаг успешно завершён (перестроение на этом закончится)  
`false`, если необходимо повторить этот шаг (в следующий раз выполнится опять эта функция)

#### 7.15.3.34 `LineChanging_()`

```
void rtm::VehicleObject::LineChanging_ (  
    WorldController *const world ) [private]
```

Функция, выполняющая различные этапы при перестроении

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

#### 7.15.3.35 `Rotation_()`

```
void rtm::VehicleObject::Rotation_ (  
    WorldController *const world ) [private]
```

Функция, выполняющая различные этапы при повороте

Аргументы

<code>world</code>	контроллер мира, в котором находится данный объект
--------------------	--

### 7.15.3.36 Movement\_()

```
void rtm::VehicleObject::Movement_ (  
    WorldController *const world ) [private]
```

Функция, выполняющая различные этапы при движении

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

### 7.15.3.37 SpeedChanging\_()

```
void rtm::VehicleObject::SpeedChanging_ (  
    WorldController *const world ) [private]
```

Функция, выполняющая изменение скорости в зависимости от ускорения, тормозного пути и т.д.

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

### 7.15.3.38 SmoothBrakingCounter()

```
void rtm::VehicleObject::SmoothBrakingCounter (  
    WorldController *const world ) [private]
```

Функция, выполняющая декрементирование тормозного пути (если задан)

Аргументы

world	контроллер мира, в котором находится данный объект
-------	--

Объявления и описания членов классов находятся в файлах:

- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/VehicleObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/VehicleObject.cpp

## 7.16 Класс rtm::WorldController

Класс контроллера карты, связующее звено всех объектов

```
#include <WorldController.h>
```

### Открытые члены

- [WorldController](#) ()  
Конструктор по умолчанию
- [WorldController](#) ([WorldScene](#) \*const scene)
- [WorldController](#) ([WorldScene](#) \*const scene, std::string const &filename)
- [WorldController](#) ([WorldScene](#) \*const scene, size\_t mapNumber)
- void [Update](#) (float time)
- cocos2d::Layer \* [GetLayer](#) () const
- size\_t [GetColumnsCount](#) () const
- size\_t [GetRowsCount](#) () const
- float [GetDeltaTime](#) () const
- float [GetTimeFactor](#) () const
- [CoatingObject](#) \* [GetCoatingObject](#) (int column, int row)
- [CoatingUnion](#) \* [GetCoatingUnion](#) (int column, int row)
- [StaticObject](#) \* [GetStaticObject](#) (int column, int row)
- std::vector< [DynamicShared](#) > & [GetDynamicObjects](#) ()
- bool [IsPause](#) ()
- bool [IsCorrectColumn](#) (int column)
- bool [IsCorrectRow](#) (int row)
- bool [IsAllowableColumn](#) (int column)
- bool [IsAllowableRow](#) (int row)
- bool [IsVisibleColumn](#) (int column)
- bool [IsVisibleRow](#) (int row)
- void [SetTimeFactor](#) (float factor)
- bool [LoadMap](#) (std::string const &filename)
- bool [LoadMap](#) (size\_t number)
- void [SpawnCar](#) ()  
Функция для добавления машины на карту
- void [RemoveAccidents](#) ()  
Функция для удаления аварий
- void [RemoveVehicles](#) ()  
Функция для удаления всего транспорта
- void [Play](#) ()  
Функция для продолжения выполнения обновлений
- void [Pause](#) ()  
Функция для временной остановки обновлений
- void [Reset](#) ()  
Функция для перезагрузки карты

### Закрытые члены

- bool [IsEmpty](#)\_ (int column, int row, size\_t width=1, size\_t height=1)
- bool [GenerateObject](#)\_ (uint8\_t \*params, uint8\_t count)
- bool [AddCoatingUnion](#)\_ (int column, int row, [CoatingUnionShared](#) coatingUnion)
- bool [AddDriveway](#)\_ ([CoatingType](#) type, int column, int row, size\_t width, size\_t height, [AngleType](#) angle)
- bool [AddCrossroad](#)\_ ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, size\_t controlUnitType=0)

- bool [AddTCrossroad\\_](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection, size\_t controlUnitType=0)
- bool [AddLeftTurt\\_](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)
- bool [AddRightTurt\\_](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)
- bool [AddControlUnit\\_](#) ([ControlUnitShared](#) controlUnit)
- bool [AddStaticObject\\_](#) (int column, int row, [StaticShared](#) staticObject)
- bool [AddBuilding\\_](#) (size\_t type, int column, int row, float angle)
- bool [AddDynamicObject\\_](#) (int column, int row, [DynamicShared](#) dynamicObject)
- bool [AddCar\\_](#) (size\_t type, int column, int row, float angle)
- size\_t [GetVectorColumn\\_](#) (int column)
- size\_t [GetVectorRow\\_](#) (int row)
- int [GetRealColumn\\_](#) (size\_t column)
- int [GetRealRow\\_](#) (size\_t row)
- void [CloseMap\\_](#) ()  
Функция закрытия карты
- void [ClearSpawns\\_](#) ()  
Функция для очистки массива точек генерации
- void [ClearCoatingObjects\\_](#) ()  
Функция для очистки матрицы покрытий
- void [ClearControlUnits\\_](#) ()  
Функция для очистки движущихся объектов
- void [ClearStaticObjects\\_](#) ()  
Функция для очистки статических объектов
- void [ClearDynamicObjects\\_](#) ()  
Функция для очистки движущихся объектов

### Закрытые статические члены

- static std::string [GetClassFile\\_](#) (size\_t number)

### Закрытые данные

- [WorldScene](#) \* [scene\\_](#)  
Сцена, к которой привязан контроллер
- cocos2d::Layer \* [mainLayer\\_](#)  
Основной слой сцены (на нём вся движуха)
- bool [isPause\\_](#)  
Происходят ли обновления (точнее стоит ли пауза)
- uint8\_t [hiddenArea\\_](#)  
Размер скрытой зоны
- uint16\_t [columnsCount\\_](#)  
Количество колонок (включая скрытую зону)
- uint16\_t [rowsCount\\_](#)  
Количество строк (включая скрытую зону)
- [SpawnVector](#) [spawns\\_](#)  
Массив точек генерации транспорта
- float [deltaTime\\_](#)  
Последняя разница между обновлениями
- float [spawnTime\\_](#)  
Время, прошедшее с последней автоматической генерации транспорта

- float [cleanTime\\_](#)  
Время, прошедшее с последнего автоматического удаления аварий
- float [timeFactor\\_](#)  
Коэффициент ускорения времени
- std::string [lastMapFile\\_](#)  
Последняя загруженная карта (путь к файлу)
- [CoatingUnionMatrix](#) [coatingUnions\\_](#)  
Матрица покрытий
- [ControlUnitVector](#) [controlUnits\\_](#)  
Матрица объединений покрытий
- [StaticMatrix](#) [staticObjects\\_](#)  
Матрица статических объектов
- [DynamicVector](#) [dynamicObjects\\_](#)  
Массив движущихся объектов

### 7.16.1 Подробное описание

Класс контроллера карты, связующее звено всех объектов

### 7.16.2 Конструктор(ы)

#### 7.16.2.1 WorldController() [1/3]

```
rtm::WorldController::WorldController (
    WorldScene *const scene )
```

Конструктор без загрузки какой-либо карты

Аргументы

scene	сцена, к которой привязан контроллер
-------	--------------------------------------

#### 7.16.2.2 WorldController() [2/3]

```
rtm::WorldController::WorldController (
    WorldScene *const scene,
    std::string const & filename )
```

Конструктор с загрузкой карты

Аргументы

scene	сцена, к которой привязан контроллер
filename	путь к карте

### 7.16.2.3 WorldController() [3/3]

```
rtm::WorldController::WorldController (  
    WorldScene *const scene,  
    size_t mapNumber )
```

Конструктор с загрузкой карты

Аргументы

scene	сцена, к которой привязан контроллер
mapNumber	номер стандартной карты

### 7.16.3 Методы

#### 7.16.3.1 Update()

```
void rtm::WorldController::Update (  
    float time )
```

Функция обновления

Аргументы

time	время, прошедшее с момента прошлого обновления (в секундах)
------	---

#### 7.16.3.2 GetLayer()

```
cocos2d::Layer * rtm::WorldController::GetLayer ( ) const
```

Функция для получения основного слоя сцены (на нём вся движуха)

Возвращает

основной слой

## 7.16.3.3 GetColumnsCount()

```
size_t rtm::WorldController::GetColumnsCount ( ) const
```

Функция для получения количества столбцов карты

Возвращает

количество столбцов

## 7.16.3.4 GetRowCount()

```
size_t rtm::WorldController::GetRowCount ( ) const
```

Функция для получения количества строк карты

Возвращает

количество строк

## 7.16.3.5 GetDeltaTime()

```
float rtm::WorldController::GetDeltaTime ( ) const
```

Функция для получения последней разницей между обновлениями

Возвращает

разница во времени между обновлениями (в секундах)

## 7.16.3.6 GetTimeFactor()

```
float rtm::WorldController::GetTimeFactor ( ) const
```

Функция для получения коэффициента ускорения времени

Возвращает

коэффициент ускорения времени (1 - реальная скорость)

## 7.16.3.7 GetCoatingObject()

```
rtm::CoatingObject * rtm::WorldController::GetCoatingObject (
    int column,
    int row )
```

Функция для получения объекта в определенной клетке

Возвращает

указатель на объект объекта

## 7.16.3.8 GetCoatingUnion()

```
rtm::CoatingUnion * rtm::WorldController::GetCoatingUnion (
    int column,
    int row )
```

Функция для получения объединения покрытий в определенной клетке

Возвращает

указатель на объект объединения покрытий

## 7.16.3.9 GetStaticObject()

```
rtm::StaticObject * rtm::WorldController::GetStaticObject (
    int column,
    int row )
```

Функция для получения статического объекта в определенной клетке

Возвращает

указатель на статический объект

## 7.16.3.10 GetDynamicObjects()

```
std::vector< rtm::DynamicShared > & rtm::WorldController::GetDynamicObjects ( )
```

Функция для получения массива движущихся объектов

Возвращает

массив движущихся объектов



7.16.3.11 `IsPause()`

```
bool rtm::WorldController::IsPause ( )
```

Функция сообщает, происходят ли обновления

Возвращает

`true`, если происходят не происходят (стоит пауза), иначе `false`

7.16.3.12 `IsCorrectColumn()`

```
bool rtm::WorldController::IsCorrectColumn (
    int column )
```

Функция проверяет корректность столбца

Аргументы

<code>column</code>	номер проверяемой колонки
---------------------	---------------------------

Возвращает

`true`, если столбец корректный, иначе `false`

7.16.3.13 `IsCorrectRow()`

```
bool rtm::WorldController::IsCorrectRow (
    int row )
```

Функция проверяет корректность строки

Аргументы

<code>row</code>	номер проверяемой строки
------------------	--------------------------

Возвращает

`true`, если строка корректная, иначе `false`

## 7.16.3.14 IsAllowableColumn()

```
bool rtm::WorldController::IsAllowableColumn (  
    int column )
```

Функция проверяет, можно ли двигаться в столбце

Аргументы

column	номер проверяемой колонки
--------	---------------------------

Возвращает

true, если в столбце можно двигаться, иначе false

## 7.16.3.15 IsAllowableRow()

```
bool rtm::WorldController::IsAllowableRow (  
    int row )
```

Функция проверяет, можно ли двигаться в строке

Аргументы

row	номер проверяемой строки
-----	--------------------------

Возвращает

true, если в строке можно двигаться, иначе false

## 7.16.3.16 IsVisibleColumn()

```
bool rtm::WorldController::IsVisibleColumn (  
    int column )
```

Функция проверяет видимость столбца

Аргументы

column	номер проверяемой колонки
--------	---------------------------

Возвращает

true, если столбец виден, иначе false

#### 7.16.3.17 isVisibleRow()

```
bool rtm::WorldController::isVisibleRow (
    int row )
```

Функция проверяет видимость строки

Аргументы

row	номер проверяемой строки
-----	--------------------------

Возвращает

true, если строка видна, иначе false

#### 7.16.3.18 setTimeFactor()

```
void rtm::WorldController::setTimeFactor (
    float factor )
```

Функция для установки коэффициента ускорения времени

Аргументы

factor	коэффициент ускорения времени (1 - реальная скорость)
--------	---

#### 7.16.3.19 loadMap() [1/2]

```
bool rtm::WorldController::loadMap (
    std::string const & filename )
```

Функция для загрузки карты из файла

Аргументы

filename	полный путь к файлу с картой
----------	------------------------------

## 7.16.3.20 LoadMap() [2/2]

```
bool rtm::WorldController::LoadMap (
    size_t number )
```

Функция для загрузки карты по номеру

Аргументы

number	номер стандартной карты
--------	-------------------------

## 7.16.3.21 IsEmpty\_()

```
bool rtm::WorldController::IsEmpty_ (
    int column,
    int row,
    size_t width = 1,
    size_t height = 1 ) [private]
```

Функция проверяет доступность зоны для генерации статических объектов и объектов объекта

Аргументы

column	левая колонка проверяемой зоны
row	нижняя строка проверяемой зоны
width	ширина проверяемой зоны
height	высота проверяемой зоны

Возвращает

true, если можно сгенерировать объект в данной зоне, иначе false

## 7.16.3.22 GenerateObject\_()

```
bool rtm::WorldController::GenerateObject_ (
    uint8_t * params,
    uint8_t count ) [private]
```

Функция для парсинга параметров и генерации объектов

## Аргументы

params	массив параметров генерации
count	количество параметров генерации в массиве

## Возвращает

true, если объект получилось сгенерировать, иначе false

## 7.16.3.23 AddCoatingUnion\_()

```
bool rtm::WorldController::AddCoatingUnion_ (
    int column,
    int row,
    CoatingUnionShared coatingUnion ) [private]
```

## Функция для генерации объединения покрытий

## Аргументы

column	левая колонка объединения покрытий
row	нижняя строка объединения покрытий
coatingUnion	умный указатель на объединение дорог

## Возвращает

true, если объект получилось сгенерировать, иначе false

## 7.16.3.24 AddDriveway\_()

```
bool rtm::WorldController::AddDriveway_ (
    CoatingType type,
    int column,
    int row,
    size_t width,
    size_t height,
    AngleType angle ) [private]
```

## Функция для генерации прямой односторонней дороги

## Аргументы

type	тип объекта (асфальт, грязь...)
column	левая колонка объекта дороги
row	нижняя строка объекта дороги
width	ширина объекта дороги
height	высота объекта дороги
angle	направление, в котором разрешено движение

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.25 AddCrossroad\_()

```
bool rtm::WorldController::AddCrossroad_ (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts,
    size_t controlUnitType = 0 ) [private]
```

Функция для генерации перекрестка

Аргументы

type	тип объекта (асфальт, грязь...)
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении
controlUnitType	тип управляющего модуля перекрестком (тип светофора)

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.26 AddTCrossroad\_()

```
bool rtm::WorldController::AddTCrossroad_ (
    CoatingType type,
    int column,
    int row,
    LinesCounts linesCounts,
    AngleType nullDirection,
    size_t controlUnitType = 0 ) [private]
```

Функция для генерации т-образного перекрестка

Аргументы

type	тип объекта (асфальт, грязь...)
column	левая колонка перекрестка
row	нижняя строка перекрестка
linesCounts	количество полос в каждом направлении
nullDirection	сторона, в направлении которой нельзя двигаться
controlUnitType	тип управляющего модуля перекрестком (тип светофора)

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.27 AddLeftTurt\_()

```
bool rtm::WorldController::AddLeftTurt_ (
    CoatingType type,
    int column,
    int row,
    size_t linesCount,
    AngleType angle ) [private]
```

Функция для генерации левого поворота

Аргументы

type	тип объекта (асфальт, грязь...)
column	левая колонка поворота
row	нижняя строка поворота
linesCount	количество полос
angle	угол поворота (самого поворота)

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.28 AddRightTurt\_()

```
bool rtm::WorldController::AddRightTurt_ (
    CoatingType type,
    int column,
    int row,
    size_t linesCount,
    AngleType angle ) [private]
```

Функция для генерации правого поворота

Аргументы

type	тип объекта (асфальт, грязь...)
column	левая колонка поворота
row	нижняя строка поворота
linesCount	количество полос
angle	угол поворота (самого поворота)

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.29 AddControlUnit\_()

```
bool rtm::WorldController::AddControlUnit_ (
    ControlUnitShared controlUnit ) [private]
```

Функция для добавления управляющего блока в общий массив (для обновлений)

Аргументы

controlUnit	умный указатель на управляющий блок
-------------	-------------------------------------

Возвращает

true, если объект получилось добавить, иначе false

#### 7.16.3.30 AddStaticObject\_()

```
bool rtm::WorldController::AddStaticObject_ (
    int column,
    int row,
    StaticShared staticObject ) [private]
```

Функция для генерации статического объекта

Аргументы

column	колонка статического объекта
row	строка статического объекта
staticObject	умный указатель на статический объект

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.31 AddBuilding\_()

```
bool rtm::WorldController::AddBuilding_ (
    size_t type,
```



```
int column,  
int row,  
float angle ) [private]
```

Функция для генерации строения

Аргументы

type	тип строения
column	колонка строения
row	строка строения
angle	угол поворота строения

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.32 AddDynamicObject\_()

```
bool rtm::WorldController::AddDynamicObject_ (  
    int column,  
    int row,  
    DynamicShared dynamicObject ) [private]
```

Функция для генерации динамического объекта

Аргументы

column	колонка динамического объекта
row	строка динамического объекта
dynamicObject	умный указатель на динамический объект

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.16.3.33 AddCar\_()

```
bool rtm::WorldController::AddCar_ (  
    size_t type,  
    int column,  
    int row,  
    float angle ) [private]
```

Функция для генерации машины

## Аргументы

type	тип машины
column	колонка машины
row	строка машины
angle	угол поворота машины

## Возвращает

true, если объект получилось сгенерировать, иначе false

## 7.16.3.34 GetVectorColumn\_()

```
size_t rtm::WorldController::GetVectorColumn_(  
    int column ) [inline], [private]
```

Функция для получения столбца в массиве

## Аргументы

column	столбец объекта
--------	-----------------

## Возвращает

столбец в массиве

## 7.16.3.35 GetVectorRow\_()

```
size_t rtm::WorldController::GetVectorRow_(  
    int row ) [inline], [private]
```

Функция для получения строки в массиве

## Аргументы

row	строка объекта
-----	----------------

## Возвращает

строка в массиве

## 7.16.3.36 GetRealColumn\_()

```
int rtm::WorldController::GetRealColumn_ (
    size_t column ) [inline], [private]
```

Функция для получения столбца объекта

Аргументы

column	столбец в массиве
--------	-------------------

Возвращает

столбец объекта

## 7.16.3.37 GetRealRow\_()

```
int rtm::WorldController::GetRealRow_ (
    size_t row ) [inline], [private]
```

Функция для получения строки объекта

Аргументы

row	строка в массиве
-----	------------------

Возвращает

строка объекта

## 7.16.3.38 GetClassFile\_()

```
std::string rtm::WorldController::GetClassFile_ (
    size_t number ) [static], [private]
```

Функция для получения файла карты по номеру

Аргументы

number	номер стандартной карты
--------	-------------------------

Возвращает

путь к файлу карты

Объявления и описания членов классов находятся в файлах:

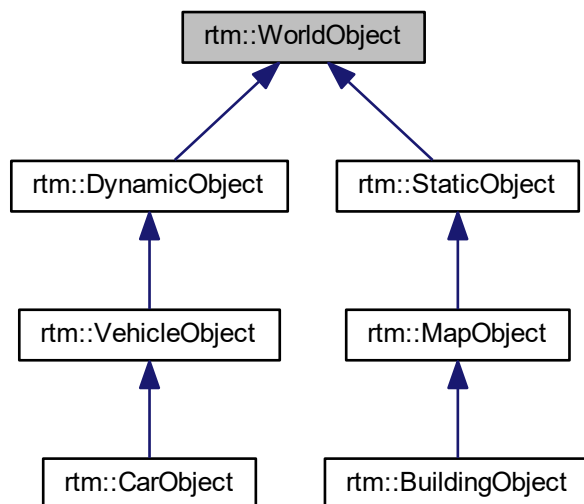
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldController.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldController.cpp

## 7.17 Класс rtm::WorldObject

Класс объекта мира (родитель всех условно объемных объектов)

```
#include <WorldObject.h>
```

Граф наследования: rtm::WorldObject:



Открытые члены

- [WorldObject](#) ()  
Конструктор по умолчанию
- [WorldObject](#) (cocos2d::Sprite \*const sprite, float x, float y, float angle)
- [WorldObject](#) (std::string const &filename, float x, float y, float angle)
- virtual [~WorldObject](#) ()=default  
Деструктор по умолчанию
- cocos2d::Sprite \* [GetSprite](#) () const
- float [GetX](#) () const
- float [GetY](#) () const
- float [GetAngle](#) () const
- float [GetWidth](#) () const
- float [GetHeight](#) () const

## Защищенные члены

- void `SetSprite_` (`cocos2d::Sprite *const sprite`)
- void `SetX_` (`float x`)
- void `SetY_` (`float y`)
- void `SetAngle_` (`float angle`)
- void `SetWidth_` (`float width`)
- void `SetHeight_` (`float height`)
- virtual void `PositionInit_` ()  
Функция, выполняемая во время инициализации
- virtual void `PositionUpdate_` ()  
Функция, выполняемая во время обновления положения
- virtual void `OnXUpdate_` ()  
Функция, выполняемая во время обновления абсциссы
- virtual void `OnYUpdate_` ()  
Функция, выполняемая во время обновления ординаты
- virtual void `OnAngleUpdate_` ()  
Функция, выполняемая во время обновления угла поворота
- virtual void `OnWidthUpdate_` ()  
Функция, выполняемая во время обновления ширины
- virtual void `OnHeightUpdate_` ()  
Функция, выполняемая во время обновления высоты

## Закрытые члены

- void `SetSpriteX_` (`float x`)
- void `SetSpriteY_` (`float y`)
- void `SetSpriteAngle_` (`float angle`)
- void `SetSpriteWidth_` (`float width`)
- void `SetSpriteHeight_` (`float height`)

## Закрытые данные

- `cocos2d::Sprite * sprite_`  
Указатель на спрайт
- float `x_`  
Абсцисса
- float `prevX_`  
Абсцисса для отслеживания изменений
- float `y_`  
Ордината
- float `prevY_`  
Ордината для отслеживания изменений
- float `angle_`  
Угол поворота
- float `prevAngle_`  
Угол поворота для отслеживания изменений
- float `width_`  
Ширина
- float `prevWidth_`  
Ширина для отслеживания изменений
- float `height_`  
Высота
- float `prevHeight_`  
Высота для отслеживания изменений

### 7.17.1 Подробное описание

Класс объекта мира (родитель всех условно объемных объектов)

### 7.17.2 Конструктор(ы)

#### 7.17.2.1 WorldObject() [1/2]

```
rtm::WorldObject::WorldObject (
    cocos2d::Sprite *const sprite,
    float x,
    float y,
    float angle )
```

Конструктор с использованием уже готового спрайта

Аргументы

sprite	указатель на готовый спрайт
x	абсцисса
y	ордината
angle	угол поворота объекта

#### 7.17.2.2 WorldObject() [2/2]

```
rtm::WorldObject::WorldObject (
    std::string const & filename,
    float x,
    float y,
    float angle )
```

Конструктор из файла

Аргументы

filename	путь к файлу инициализации
x	абсцисса
y	ордината
angle	угол поворота объекта

### 7.17.3 Методы

## 7.17.3.1 GetSprite()

```
cocos2d::Sprite * rtm::WorldObject::GetSprite ( ) const
```

Функция для получения спрайта

Возвращает

указатель на спрайт

## 7.17.3.2 GetX\_()

```
float rtm::WorldObject::GetX_ ( ) const
```

Функция для получения абсциссы

Возвращает

абсцисса

## 7.17.3.3 GetY\_()

```
float rtm::WorldObject::GetY_ ( ) const
```

Функция для получения ординаты

Возвращает

ордината

## 7.17.3.4 GetAngle()

```
float rtm::WorldObject::GetAngle ( ) const
```

Функция для получения угла поворота

Возвращает

угол поворота

#### 7.17.3.5 GetWidth()

```
float rtm::WorldObject::GetWidth ( ) const
```

Функция для получения ширины

Возвращает

ширина

#### 7.17.3.6 GetHeight()

```
float rtm::WorldObject::GetHeight ( ) const
```

Функция для получения высоты

Возвращает

высота

#### 7.17.3.7 SetSprite\_()

```
void rtm::WorldObject::SetSprite_ (
    cocos2d::Sprite *const sprite ) [protected]
```

Функция для установки спрайта

Аргументы

sprite	указатель на спрайт
--------	---------------------

#### 7.17.3.8 SetX\_()

```
void rtm::WorldObject::SetX_ (
    float x ) [protected]
```

Функция для установки абсциссы

Аргументы

x	абсцисса
---	----------



## 7.17.3.9 SetY\_()

```
void rtm::WorldObject::SetY_ (
    float y ) [protected]
```

Функция для установки ординаты

Аргументы

y	ордината
---	----------

## 7.17.3.10 SetAngle\_()

```
void rtm::WorldObject::SetAngle_ (
    float angle ) [protected]
```

Функция для установки угла поворота

Аргументы

angle	угол поворота
-------	---------------

## 7.17.3.11 SetWidth\_()

```
void rtm::WorldObject::SetWidth_ (
    float width ) [protected]
```

Функция для установки ширины

Аргументы

width	ширина
-------	--------

## 7.17.3.12 SetHeight\_()

```
void rtm::WorldObject::SetHeight_ (
    float height ) [protected]
```

Функция для установки высоты

Аргументы

height	высота
--------	--------

#### 7.17.3.13 SetSpriteX\_()

```
void rtm::WorldObject::SetSpriteX_ (
    float x ) [private]
```

Функция для установки абсциссы спрайта

Аргументы

x	абсцисса спрайта
---	------------------

#### 7.17.3.14 SetSpriteY\_()

```
void rtm::WorldObject::SetSpriteY_ (
    float y ) [private]
```

Функция для установки ординаты спрайта

Аргументы

y	ордината спрайта
---	------------------

#### 7.17.3.15 SetSpriteAngle\_()

```
void rtm::WorldObject::SetSpriteAngle_ (
    float angle ) [private]
```

Функция для установки угла поворота спрайта

Аргументы

angle	угол поворота спрайта
-------	-----------------------

## 7.17.3.16 SetSpriteWidth\_()

```
void rtm::WorldObject::SetSpriteWidth_ (
    float width ) [private]
```

Функция для установки ширины спрайта

Аргументы

width	ширина спрайта
-------	----------------

## 7.17.3.17 SetSpriteHeight\_()

```
void rtm::WorldObject::SetSpriteHeight_ (
    float height ) [private]
```

Функция для установки высоты спрайта

Аргументы

height	высота спрайта
--------	----------------

Объявления и описания членов классов находятся в файлах:

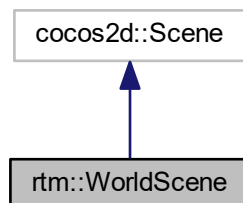
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldObject.cpp

## 7.18 Класс rtm::WorldScene

Класс главной сцены, на которой всё и происходит (для отрисовки)

```
#include <WorldScene.h>
```

Граф наследования:rtm::WorldScene:



## Открытые члены

- `~WorldScene ()=default`  
Деструктор по умолчанию
- `virtual bool init () override`
- `virtual void update (float time) override`
- `cocos2d::Layer * GetMainLayer () const`

## Функции для установки фона

- `void SetBackground (std::string const &filename)`
- `void SetBackground (size_t number)`

## Открытые статические члены

- `static WorldScene * Create ()`

## Закрытые члены

- `void OpenMap_ ()`  
Функция открытия карты
- `void Restart_ ()`  
Функция перезагрузки карты
- `void SetDefaultPosition_ ()`  
Функция для установки первоначальной позиции просмотра
- `void ShiftUp_ ()`  
Функция сдвига области просмотра вверх
- `void ShiftRight_ ()`  
Функция сдвига области просмотра вправо
- `void ShiftDown_ ()`  
Функция сдвига области просмотра вниз
- `void ShiftLeft_ ()`  
Функция сдвига области просмотра влево
- `void UpdatePosition_ ()`  
Функция для обновления положения главного слоя в зависимости от области просмотра
- `void SetDefaultScale_ ()`  
Функция для установки масштаба просмотра по умолчанию
- `void IncreaseScale_ ()`  
Функция для увеличения масштаба просмотра
- `void DecreaseScale_ ()`  
Функция для уменьшения масштаба просмотра
- `void SetDefaultSpeed_ ()`  
Функция для установки скорости обработки (скорости объектов) по умолчанию
- `void IncreaseSpeed_ ()`  
Функция для увеличения скорости обработки (скорости объектов)
- `void DecreaseSpeed_ ()`  
Функция для уменьшения скорости обработки (скорости объектов)
- `WorldControllerUnique & GetMap_ ()`

### Закрытые статические члены

- static `std::string` `GetBackgroundFile_` (`size_t` number)
- static void `KeyPressed_` (`cocos2d::EventKeyboard::KeyCode` code, `cocos2d::Event *event`)  
Функция-обработчик нажатий клавиш клавиатуры
- static void `KeyReleased_` (`cocos2d::EventKeyboard::KeyCode` code, `cocos2d::Event *event`)  
Функция-обработчик отпусканий клавиш клавиатуры

### Закрытые данные

- `cocos2d::Layer *` `mainLayer_`  
Основной слой, на котором располагаются объекты
- `cocos2d::Layer *` `backgroundLayer_`  
Слой для фона, находится позади основного
- `cocos2d::Sprite *` `background_`  
Картинка фона
- `WorldControllerUnique` `map_`  
Контроллер мира, привязанный к данной сцене
- float `clickTime_`  
Время, прошедшее с последнего нажатия клавиш перемещения по карте (стрелочек)
- int `viewColumn_`  
Сдвиг по горизонтали при просмотре
- int `viewRow_`  
Сдвиг по вертикали при просмотре
- bool `isCtrlPressed_`  
Состояние клавиши CTRL.
- bool `isAltPressed_`  
Состояние клавиши ALT.
- bool `isUpArrowPressed_`  
Состояние клавиши "вверх".
- bool `isRightArrowPressed_`  
Состояние клавиши "вправо".
- bool `isDownArrowPressed_`  
Состояние клавиши "вниз".
- bool `isLeftArrowPressed_`  
Состояние клавиши "влево".

### Закрытые статические данные

- static `WorldScene *` `globalScene_` { `nullptr` }  
Основная сцена, к которой адресуются нажатия клавиш и т.д.

#### 7.18.1 Подробное описание

Класс главной сцены, на которой всё и происходит (для отрисовки)

#### 7.18.2 Методы

#### 7.18.2.1 Create()

```
rtm::WorldScene * rtm::WorldScene::Create ( ) [static]
```

Конструктор класса, поддерживающий RAII

Возвращает

указатель на созданный объект

#### 7.18.2.2 init()

```
bool rtm::WorldScene::init ( ) [override], [virtual]
```

Функция для инициализации полей

Возвращает

true в случае успешной инициализации, иначе false

#### 7.18.2.3 update()

```
void rtm::WorldScene::update (
    float time ) [override], [virtual]
```

Функция для обновления сцены

Аргументы

time	время, прошедшее с момента прошлого обновления
------	--

#### 7.18.2.4 GetMainLayer()

```
cocos2d::Layer * rtm::WorldScene::GetMainLayer ( ) const
```

Функция для получения основного слоя, на котором находятся объекты

Возвращает

основной слой

## 7.18.2.5 SetBackground() [1/2]

```
void rtm::WorldScene::SetBackground (
    std::string const & filename )
```

Функции для установки фона из файла

Аргументы

filename	полный путь к файлу с фоном (картинка)
----------	--

## 7.18.2.6 SetBackground() [2/2]

```
void rtm::WorldScene::SetBackground (
    size_t number )
```

Функции для установки фона по номеру

Аргументы

number	номер стандартного фона
--------	-------------------------

## 7.18.2.7 GetMap\_()

```
rtm::WorldControllerUnique & rtm::WorldScene::GetMap_ ( ) [private]
```

Функция для получения контроллера данной сцены

Возвращает

контроллер сцены

## 7.18.2.8 GetBackgroundFile\_()

```
std::string rtm::WorldScene::GetBackgroundFile_ (
    size_t number ) [static], [private]
```

Функция для получения файла фона по номеру

Аргументы

number	номер стандартного фона
--------	-------------------------

Возвращает

путь к файлу фона

Объявления и описания членов классов находятся в файлах:

- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldScene.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldScene.cpp



# Предметный указатель

- AddBuilding\_
  - rtm::WorldController, [114](#)
- AddCar\_
  - rtm::WorldController, [115](#)
- AddCoatingUnion\_
  - rtm::WorldController, [111](#)
- AddControlUnit\_
  - rtm::WorldController, [114](#)
- AddCrossroad\_
  - rtm::WorldController, [112](#)
- AddDriveway\_
  - rtm::WorldController, [111](#)
- AddDynamicObject\_
  - rtm::WorldController, [115](#)
- AddLeftTurt\_
  - rtm::WorldController, [113](#)
- AddRightTurt\_
  - rtm::WorldController, [113](#)
- AddStaticObject\_
  - rtm::WorldController, [114](#)
- AddTCrossroad\_
  - rtm::WorldController, [112](#)
- AfterMoving\_
  - rtm::VehicleObject, [95](#)
- Allowed
  - rtm, [20](#)
- AngleToAngleType
  - rtm, [26](#)
- AngleToDirection
  - rtm, [26](#)
- AngleType
  - rtm, [18](#)
- AngleTypeToAngle
  - rtm, [27](#)
- AngleTypeToDirection
  - rtm, [27](#)
- AppDelegate, [33](#)
  - applicationDidFinishLaunching, [34](#)
- applicationDidFinishLaunching
  - AppDelegate, [34](#)
- BeforeMoving\_
  - rtm::VehicleObject, [95](#)
- BuildingObject
  - rtm::BuildingObject, [35, 36](#)
- CARS\_ACCELERATIONS
  - rtm, [31](#)
- CARS\_MAX\_SPEEDS
  - rtm, [30](#)
- CarObject
  - rtm::CarObject, [38, 39](#)
- CellToPixel
  - rtm, [25](#)
- CenterIsCrossed
  - rtm, [23](#)
- ChangeLine\_
  - rtm::VehicleObject, [89](#)
- CheckCoatingAhead\_
  - rtm::CarObject, [41](#)
- CheckCoatingUnionAhead\_
  - rtm::CarObject, [41](#)
- CheckCollisions
  - rtm, [21](#)
  - rtm::DynamicObject, [73](#)
- CheckCrossroadArea\_
  - rtm::VehicleObject, [94](#)
- CheckForwardArea\_
  - rtm::VehicleObject, [93](#)
- CheckForwardCoating\_
  - rtm::VehicleObject, [92](#)
- CheckForwardCoatingUnion\_
  - rtm::VehicleObject, [92](#)
- CheckLineChangingArea\_
  - rtm::VehicleObject, [95](#)
- CheckMovingArea\_
  - rtm::VehicleObject, [93](#)
- CheckRoadAhead\_
  - rtm::CarObject, [42](#)
- CheckRotationArea\_
  - rtm::VehicleObject, [94](#)
- CheckTurnArea\_
  - rtm::VehicleObject, [93](#)
- Closed
  - rtm, [20](#)
- CoatingObject
  - rtm::CoatingObject, [44, 45](#)
- CoatingType
  - rtm, [20](#)
- CoatingUnion
  - rtm::CoatingUnion, [49](#)
- CoatingUnionType
  - rtm, [19](#)
- ControlUnit
  - rtm::ControlUnit, [54](#)
- CountDeceleration
  - rtm, [28](#)
- CountLength\_
  - rtm, [30](#)

- rtm::DrivewayObject, [66](#)
- CountLines\_
  - rtm::DrivewayObject, [67](#)
- Create
  - rtm::WorldScene, [127](#)
- CrossroadMatrix
  - rtm::CrossroadObject, [59](#)
- CrossroadObject
  - rtm::CrossroadObject, [59](#)
- CrossroadSignals
  - rtm, [17](#)
- DEFAULT\_CROSSROAD\_SIGNALS
  - rtm, [29](#)
- DEFAULT DIRECTIONS\_SIGNAL\_SPRITES
  - rtm, [29](#)
- DirectionSignalIndex
  - rtm, [19](#)
- DirectionSignals
  - rtm, [17](#)
- DirectionToAngle
  - rtm, [27](#)
- DirectionToAngleType
  - rtm, [28](#)
- DirectionType
  - rtm, [19](#)
- Directions
  - rtm, [17](#)
- DirectionsSignalSprites
  - rtm, [18](#)
- DistanceToNextCenter
  - rtm, [23](#)
- Down
  - rtm, [19](#)
- Downward
  - rtm, [19](#)
- DrivewayMatrix
  - rtm::DrivewayObject, [64](#)
- DrivewayObject
  - rtm::DrivewayObject, [63](#)
- DynamicObject
  - rtm::DynamicObject, [69](#)
- Forbidden
  - rtm, [20](#)
- GenerateObject\_
  - rtm::WorldController, [110](#)
- GetAngle
  - rtm::TurnObject, [84](#)
  - rtm::WorldObject, [121](#)
- GetBackgroundFile\_
  - rtm::WorldScene, [129](#)
- GetClassAcceleration\_
  - rtm::CarObject, [43](#)
- GetClassDirections\_
  - rtm::RoadCoating, [78](#)
- GetClassFile\_
  - rtm::BuildingObject, [36](#)
- rtm::CarObject, [42](#)
- rtm::RoadCoating, [78](#)
- rtm::WorldController, [117](#)
- GetClassMaxSpeed\_
  - rtm::CarObject, [42](#)
- GetClassResistance\_
  - rtm::RoadCoating, [78](#)
- GetCoatingObject
  - rtm::CoatingUnion, [50](#)
  - rtm::WorldController, [105](#)
- GetCoatingUnion
  - rtm::WorldController, [106](#)
- GetColumn\_
  - rtm::CoatingUnion, [52](#)
- GetColumnsCount
  - rtm::WorldController, [104](#)
- GetControlUnit
  - rtm::CrossroadObject, [61](#)
- GetDeltaTime
  - rtm::WorldController, [105](#)
- GetDynamicObjects
  - rtm::WorldController, [106](#)
- GetFinalSpeed\_
  - rtm::VehicleObject, [91](#)
- GetHeight
  - rtm::CoatingUnion, [50](#)
  - rtm::WorldObject, [122](#)
- GetLastDelta
  - rtm::DynamicObject, [70](#)
- GetLayer
  - rtm::WorldController, [104](#)
- GetLength
  - rtm::CoatingUnion, [51](#)
  - rtm::DrivewayObject, [64](#)
- GetLinesCount
  - rtm::DrivewayObject, [64](#)
- GetMainLayer
  - rtm::WorldScene, [128](#)
- GetMap\_
  - rtm::WorldScene, [129](#)
- GetMaxSpeed\_
  - rtm::VehicleObject, [90](#)
- GetNullDirection
  - rtm::CrossroadObject, [60](#)
- GetRealColumn\_
  - rtm::WorldController, [116](#)
- GetRealRow\_
  - rtm::WorldController, [117](#)
- GetResistance
  - rtm::CoatingObject, [46](#)
- GetRow\_
  - rtm::CoatingUnion, [52](#)
- GetRowsCount
  - rtm::WorldController, [105](#)
- GetSignal
  - rtm::ControlUnit, [55](#)
- GetSignalFile\_
  - rtm::ControlUnit, [57](#)

- GetSpeed
  - rtm::DynamicObject, [70](#)
- GetSprite
  - rtm::CoatingObject, [45](#)
  - rtm::WorldObject, [120](#)
- GetStaticObject
  - rtm::WorldController, [106](#)
- GetTimeFactor
  - rtm::WorldController, [105](#)
- GetType
  - rtm::CoatingUnion, [50](#)
- GetVectorColumn\_
  - rtm::WorldController, [116](#)
- GetVectorRow\_
  - rtm::WorldController, [116](#)
- GetWidth
  - rtm::CoatingUnion, [50](#)
  - rtm::WorldObject, [121](#)
- GetX\_
  - rtm::WorldObject, [121](#)
- GetY\_
  - rtm::WorldObject, [121](#)
- HasCollision
  - rtm::DynamicObject, [70](#)
- HasDirection
  - rtm::CoatingObject, [46](#)
- InCenter
  - rtm, [23](#)
- init
  - rtm::WorldScene, [128](#)
- IsAllowableColumn
  - rtm::WorldController, [107](#)
- IsAllowableRow
  - rtm::WorldController, [108](#)
- IsBeholding\_
  - rtm::DynamicObject, [72](#)
- IsBraking\_
  - rtm::VehicleObject, [90](#)
- IsCorrectColumn
  - rtm::CoatingUnion, [51](#)
  - rtm::WorldController, [107](#)
- IsCorrectRow
  - rtm::CoatingUnion, [51](#)
  - rtm::WorldController, [107](#)
- IsDirectionAvailable
  - rtm::CoatingObject, [46](#)
- IsEmpty\_
  - rtm::WorldController, [110](#)
- IsIntersecting\_
  - rtm::DynamicObject, [72](#)
- isLeftLine
  - rtm::DrivewayObject, [65](#), [66](#)
- IsLineChanging\_
  - rtm::VehicleObject, [90](#)
- IsMovement\_
  - rtm::VehicleObject, [89](#)
- IsNear\_
  - rtm::DynamicObject, [73](#)
- IsNearOthers
  - rtm::DynamicObject, [71](#)
- IsPause
  - rtm::WorldController, [106](#)
- IsRight
  - rtm::TurnObject, [84](#)
- isRightLine
  - rtm::DrivewayObject, [65](#)
- IsRotation\_
  - rtm::VehicleObject, [90](#)
- IsVisibleColumn
  - rtm::WorldController, [108](#)
- IsVisibleRow
  - rtm::WorldController, [109](#)
- Left
  - rtm, [19](#)
- LeftTurnMatrix
  - rtm::TurnObject, [84](#)
- Leftward
  - rtm, [19](#)
- LineChanging\_
  - rtm::VehicleObject, [99](#)
- LineChangingEnd\_
  - rtm::VehicleObject, [99](#)
- LineChangingStart
  - rtm::CarObject, [41](#)
  - rtm::VehicleObject, [98](#)
- LineChangingTick\_
  - rtm::VehicleObject, [98](#)
- LinesCounts
  - rtm, [17](#)
- LoadMap
  - rtm::WorldController, [109](#), [110](#)
- MapObject
  - rtm::MapObject, [74](#), [75](#)
- MoveForward\_
  - rtm::VehicleObject, [88](#)
- Movement\_
  - rtm::VehicleObject, [100](#)
- MovementEnd\_
  - rtm::CarObject, [40](#)
  - rtm::VehicleObject, [96](#)
- MovementStart\_
  - rtm::CarObject, [40](#)
  - rtm::VehicleObject, [96](#)
- MovementTick\_
  - rtm::CarObject, [40](#)
  - rtm::VehicleObject, [96](#)
- NEAR\_DELTA
  - rtm, [29](#)
- NormalizeAngle
  - rtm, [25](#)
- operator bool
  - rtm::ControlUnit, [55](#)

- PixelToCell
  - rtm, 25
- ROADS\_DIRECTIONS
  - rtm, 30
- ROADS\_RESISTANCES
  - rtm, 30
- ReleaseSprites
  - rtm::CoatingUnion, 52
  - rtm::ControlUnit, 56
  - rtm::CrossroadObject, 61
- Right
  - rtm, 19
- RightTurnMatrix
  - rtm::TurnObject, 83
- Rightward
  - rtm, 19
- RoadCoating
  - rtm::RoadCoating, 76, 77
- Rotate\_
  - rtm::VehicleObject, 89
- Rotation\_
  - rtm::VehicleObject, 99
- RotationEnd\_
  - rtm::VehicleObject, 97
- RotationStart\_
  - rtm::VehicleObject, 97
- RotationTick\_
  - rtm::VehicleObject, 97
- RoundAngle
  - rtm, 24
- RoundCoordinate
  - rtm, 22
- RoundToCenter
  - rtm, 22
- rtm, 11
  - Allowed, 20
  - AngleToAngleType, 26
  - AngleToDirection, 26
  - AngleType, 18
  - AngleTypeToAngle, 27
  - AngleTypeToDirection, 27
  - CARS\_ACCELERATIONS, 31
  - CARS\_MAX\_SPEEDS, 30
  - CellToPixel, 25
  - CenterIsCrossed, 23
  - CheckCollisions, 21
  - Closed, 20
  - CoatingType, 20
  - CoatingUnionType, 19
  - CountDeceleration, 28
  - CrossroadSignals, 17
  - DEFAULT\_CROSSROAD\_SIGNALS, 29
  - DEFAULT\_DIRECTIONS\_SIGNAL\_SP↔
    - RTES, 29
  - DirectionSignalIndex, 19
  - DirectionSignals, 17
  - DirectionToAngle, 27
  - DirectionToAngleType, 28
  - DirectionType, 19
  - Directions, 17
  - DirectionsSignalSprites, 18
  - DistanceToNextCenter, 23
  - Down, 19
  - Downward, 19
  - Forbidden, 20
  - InCenter, 23
  - Left, 19
  - Leftward, 19
  - LinesCounts, 17
  - NEAR\_DELTA, 29
  - NormalizeAngle, 25
  - PixelToCell, 25
  - ROADS\_DIRECTIONS, 30
  - ROADS\_RESISTANCES, 30
  - Right, 19
  - Rightward, 19
  - RoundAngle, 24
  - RoundCoordinate, 22
  - RoundToCenter, 22
  - SameAngles, 24
  - SameCoordinates, 21
  - SignalFileId, 21
  - SignalSprites, 18
  - SignalType, 20
  - SignalsSprites, 18
  - Started, 20
  - StateType, 20
  - SumAngleTypes, 28
  - Up, 19
  - Upward, 19
  - Warning, 20
- rtm::BuildingObject, 34
  - BuildingObject, 35, 36
  - GetClassFile\_, 36
- rtm::CarObject, 37
  - CarObject, 38, 39
  - CheckCoatingAhead\_, 41
  - CheckCoatingUnionAhead\_, 41
  - CheckRoadAhead\_, 42
  - GetClassAcceleration\_, 43
  - GetClassFile\_, 42
  - GetClassMaxSpeed\_, 42
  - LineChangingStart, 41
  - MovementEnd\_, 40
  - MovementStart\_, 40
  - MovementTick\_, 40
- rtm::CoatingObject, 43
  - CoatingObject, 44, 45
  - GetResistance, 46
  - GetSprite, 45
  - HasDirection, 46
  - IsDirectionAvailable, 46
  - SetDirectionAvailability, 47
  - SetSprite\_, 47
  - SetX\_, 47
  - SetY\_, 48

- rtm::CoatingUnion, 48
  - CoatingUnion, 49
  - GetCoatingObject, 50
  - GetColumn\_, 52
  - GetHeight, 50
  - GetLength, 51
  - GetRow\_, 52
  - GetType, 50
  - GetWidth, 50
  - IsCorrectColumn, 51
  - IsCorrectRow, 51
  - ReleaseSprites, 52
  - ShowSprites, 51
- rtm::ControlUnit, 53
  - ControlUnit, 54
  - GetSignal, 55
  - GetSignalFile\_, 57
  - operator bool, 55
  - ReleaseSprites, 56
  - SetState\_, 57
  - ShowSprites, 56
  - Update, 55
  - UpdateSignal\_, 56
- rtm::CrossroadObject, 57
  - CrossroadMatrix, 59
  - CrossroadObject, 59
  - GetControlUnit, 61
  - GetNullDirection, 60
  - ReleaseSprites, 61
  - ShowSprites, 61
  - TCrossroadMatrix, 60
- rtm::DrivewayObject, 62
  - CountLength\_, 66
  - CountLines\_, 67
  - DrivewayMatrix, 64
  - DrivewayObject, 63
  - GetLength, 64
  - GetLinesCount, 64
  - isLeftLine, 65, 66
  - isRightLine, 65
- rtm::DynamicObject, 67
  - CheckCollisions, 73
  - DynamicObject, 69
  - GetLastDelta, 70
  - GetSpeed, 70
  - HasCollision, 70
  - IsBeholding\_, 72
  - IsIntersecting\_, 72
  - IsNear\_, 73
  - IsNearOthers, 71
  - SetCollisionFlag\_, 72
  - SetSpeed\_, 71
  - Update, 71
- rtm::MapObject, 74
  - MapObject, 74, 75
- rtm::RoadCoating, 75
  - GetClassDirections\_, 78
  - GetClassFile\_, 78
  - GetClassResistance\_, 78
  - RoadCoating, 76, 77
- rtm::SpawnType, 79
- rtm::StaticObject, 80
  - StaticObject, 80, 81
- rtm::TurnObject, 81
  - GetAngle, 84
  - IsRight, 84
  - LeftTurnMatrix, 84
  - RightTurnMatrix, 83
  - TurnObject, 83
- rtm::VehicleObject, 85
  - AfterMoving\_, 95
  - BeforeMoving\_, 95
  - ChangeLine\_, 89
  - CheckCrossroadArea\_, 94
  - CheckForwardArea\_, 93
  - CheckForwardCoating\_, 92
  - CheckForwardCoatingUnion\_, 92
  - CheckLineChangingArea\_, 95
  - CheckMovingArea\_, 93
  - CheckRotationArea\_, 94
  - CheckTurnArea\_, 93
  - GetFinalSpeed\_, 91
  - GetMaxSpeed\_, 90
  - IsBraking\_, 90
  - IsLineChanging\_, 90
  - IsMovement\_, 89
  - IsRotation\_, 90
  - LineChanging\_, 99
  - LineChangingEnd\_, 99
  - LineChangingStart, 98
  - LineChangingTick\_, 98
  - MoveForward\_, 88
  - Movement\_, 100
  - MovementEnd\_, 96
  - MovementStart\_, 96
  - MovementTick\_, 96
  - Rotate\_, 89
  - Rotation\_, 99
  - RotationEnd\_, 97
  - RotationStart\_, 97
  - RotationTick\_, 97
  - SetBrakingFactor\_, 91
  - SetFinalSpeed\_, 91
  - SmoothBrakingCounter, 100
  - SpeedChanging\_, 100
  - Stop\_, 89
  - StopAtDistance\_, 92
  - Update, 88
  - VehicleObject, 87, 88
- rtm::WorldController, 100
  - AddBuilding\_, 114
  - AddCar\_, 115
  - AddCoatingUnion\_, 111
  - AddControlUnit\_, 114
  - AddCrossroad\_, 112
  - AddDriveway\_, 111

- AddDynamicObject\_, 115
- AddLeftTurt\_, 113
- AddRightTurt\_, 113
- AddStaticObject\_, 114
- AddTCrossroad\_, 112
- GenerateObject\_, 110
- GetClassFile\_, 117
- GetCoatingObject, 105
- GetCoatingUnion, 106
- GetColumnsCount, 104
- GetDeltaTime, 105
- GetDynamicObjects, 106
- GetLayer, 104
- GetRealColumn\_, 116
- GetRealRow\_, 117
- GetRowsCount, 105
- GetStaticObject, 106
- GetTimeFactor, 105
- GetVectorColumn\_, 116
- GetVectorRow\_, 116
- IsAllowableColumn, 107
- IsAllowableRow, 108
- IsCorrectColumn, 107
- IsCorrectRow, 107
- IsEmpty\_, 110
- IsPause, 106
- IsVisibleColumn, 108
- IsVisibleRow, 109
- LoadMap, 109, 110
- SetTimeFactor, 109
- Update, 104
- WorldController, 103, 104
- rtm::WorldObject, 118
  - GetAngle, 121
  - GetHeight, 122
  - GetSprite, 120
  - GetWidth, 121
  - GetX\_, 121
  - GetY\_, 121
  - SetAngle\_, 123
  - SetHeight\_, 123
  - SetSprite\_, 122
  - SetSpriteAngle\_, 124
  - SetSpriteHeight\_, 125
  - SetSpriteWidth\_, 124
  - SetSpriteX\_, 124
  - SetSpriteY\_, 124
  - SetWidth\_, 123
  - SetX\_, 122
  - SetY\_, 123
  - WorldObject, 120
- rtm::WorldScene, 125
  - Create, 127
  - GetBackgroundFile\_, 129
  - GetMainLayer, 128
  - GetMap\_, 129
  - init, 128
  - SetBackground, 128, 129
  - update, 128
- SameAngles
  - rtm, 24
- SameCoordinates
  - rtm, 21
- SetAngle\_
  - rtm::WorldObject, 123
- SetBackground
  - rtm::WorldScene, 128, 129
- SetBrakingFactor\_
  - rtm::VehicleObject, 91
- SetCollisionFlag\_
  - rtm::DynamicObject, 72
- SetDirectionAvailability
  - rtm::CoatingObject, 47
- SetFinalSpeed\_
  - rtm::VehicleObject, 91
- SetHeight\_
  - rtm::WorldObject, 123
- SetSpeed\_
  - rtm::DynamicObject, 71
- SetSprite\_
  - rtm::CoatingObject, 47
  - rtm::WorldObject, 122
- SetSpriteAngle\_
  - rtm::WorldObject, 124
- SetSpriteHeight\_
  - rtm::WorldObject, 125
- SetSpriteWidth\_
  - rtm::WorldObject, 124
- SetSpriteX\_
  - rtm::WorldObject, 124
- SetSpriteY\_
  - rtm::WorldObject, 124
- SetState\_
  - rtm::ControlUnit, 57
- SetTimeFactor
  - rtm::WorldController, 109
- SetWidth\_
  - rtm::WorldObject, 123
- SetX\_
  - rtm::CoatingObject, 47
  - rtm::WorldObject, 122
- SetY\_
  - rtm::CoatingObject, 48
  - rtm::WorldObject, 123
- ShowSprites
  - rtm::CoatingUnion, 51
  - rtm::ControlUnit, 56
  - rtm::CrossroadObject, 61
- SignalFileId
  - rtm, 21
- SignalSprites
  - rtm, 18
- SignalType
  - rtm, 20
- SignalsSprites
  - rtm, 18

SmoothBrakingCounter  
    rtm::VehicleObject, [100](#)  
SpeedChanging\_  
    rtm::VehicleObject, [100](#)  
Started  
    rtm, [20](#)  
StateType  
    rtm, [20](#)  
StaticObject  
    rtm::StaticObject, [80](#), [81](#)  
Stop\_  
    rtm::VehicleObject, [89](#)  
StopAtDistance\_  
    rtm::VehicleObject, [92](#)  
SumAngleTypes  
    rtm, [28](#)  
  
TCrossroadMatrix  
    rtm::CrossroadObject, [60](#)  
TurnObject  
    rtm::TurnObject, [83](#)  
  
Up  
    rtm, [19](#)  
Update  
    rtm::ControlUnit, [55](#)  
    rtm::DynamicObject, [71](#)  
    rtm::VehicleObject, [88](#)  
    rtm::WorldController, [104](#)  
update  
    rtm::WorldScene, [128](#)  
UpdateSignal\_  
    rtm::ControlUnit, [56](#)  
Upward  
    rtm, [19](#)  
  
VehicleObject  
    rtm::VehicleObject, [87](#), [88](#)  
  
Warning  
    rtm, [20](#)  
WorldController  
    rtm::WorldController, [103](#), [104](#)  
WorldObject  
    rtm::WorldObject, [120](#)