

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 . . . . .	18
6.1.2	Типы . . . . .	18
6.1.2.1	<code>Directions</code> . . . . .	18
6.1.2.2	<code>LinesCounts</code> . . . . .	18
6.1.2.3	<code>DirectionSignals</code> . . . . .	18
6.1.2.4	<code>CrossroadSignals</code> . . . . .	19
6.1.2.5	<code>SignalSprites</code> . . . . .	19
6.1.2.6	<code>SignalsSprites</code> . . . . .	19
6.1.2.7	<code>DirectionsSignalSprites</code> . . . . .	19
6.1.3	Перечисления . . . . .	19
6.1.3.1	<code>AngleType</code> . . . . .	19

6.1.3.2	DirectionType	20
6.1.3.3	CoatingUnionType	20
6.1.3.4	DirectionSignalIndex	21
6.1.3.5	SignalType	21
6.1.3.6	StateType	21
6.1.3.7	CoatingType	22
6.1.3.8	RoadType	22
6.1.3.9	SignalFileId	22
6.1.4	Функции	23
6.1.4.1	CheckCollisions()	23
6.1.4.2	SameCoordinates()	23
6.1.4.3	RoundCoordinate()	23
6.1.4.4	RoundToCenter()	24
6.1.4.5	InCenter()	24
6.1.4.6	DistanceToNextCenter()	25
6.1.4.7	CenterIsCrossed()	25
6.1.4.8	SameAngles()	25
6.1.4.9	RoundAngle()	26
6.1.4.10	NormalizeAngle()	26
6.1.4.11	PixelToCell()	27
6.1.4.12	CellToPixel()	27
6.1.4.13	AngleToAngleType()	27
6.1.4.14	AngleToDirection()	28
6.1.4.15	AngleTypeToAngle()	28
6.1.4.16	AngleTypeToDirection()	28
6.1.4.17	DirectionToAngle()	29
6.1.4.18	DirectionToAngleType()	29
6.1.4.19	GetFilename()	29
6.1.4.20	SumAngleTypes()	30
6.1.4.21	CountDeceleration()	30
6.1.5	Переменные	31
6.1.5.1	NEAR_DELTA	31
6.1.5.2	DEFAULT_CROSSROAD_SIGNALS	31
6.1.5.3	DEFAULT_DIRECTIONS_SIGNAL_SPRITES	31
6.1.5.4	COATING_INDEXES	32
6.1.5.5	ROADS_RESISTANCES	32
6.1.5.6	ROADS_DIRECTIONS	32
6.1.5.7	CARS_MAX_SPEEDS	33
6.1.5.8	CARS_ACCELERATIONS	33

7	Классы	35
7.1	Класс AppDelegate	35
7.1.1	Подробное описание	36
7.1.2	Методы	36
7.1.2.1	applicationDidFinishLaunching()	36
7.2	Класс rtm::BuildingObject	36
7.2.1	Подробное описание	37
7.2.2	Конструктор(ы)	37
7.2.2.1	BuildingObject() [1/3]	37
7.2.2.2	BuildingObject() [2/3]	37
7.2.2.3	BuildingObject() [3/3]	38
7.3	Класс rtm::BushObject	38
7.3.1	Подробное описание	39
7.3.2	Конструктор(ы)	39
7.3.2.1	BushObject() [1/3]	40
7.3.2.2	BushObject() [2/3]	40
7.3.2.3	BushObject() [3/3]	40
7.4	Класс rtm::CarObject	41
7.4.1	Подробное описание	43
7.4.2	Конструктор(ы)	43
7.4.2.1	CarObject() [1/3]	43
7.4.2.2	CarObject() [2/3]	44
7.4.2.3	CarObject() [3/3]	44
7.4.3	Методы	44
7.4.3.1	MovementStart_()	45
7.4.3.2	MovementTick_()	45
7.4.3.3	MovementEnd_()	45
7.4.3.4	LineChangingStart()	45
7.4.3.5	CheckCoatingAhead_()	47
7.4.3.6	CheckCoatingUnionAhead_()	47

7.4.3.7	CheckRoadAhead_()	47
7.4.3.8	GetClassMaxSpeed_()	48
7.4.3.9	GetClassAcceleration_()	48
7.5	Класс <code>rtm::CoatingObject</code>	48
7.5.1	Подробное описание	50
7.5.2	Конструктор(ы)	50
7.5.2.1	CoatingObject() [1/2]	50
7.5.2.2	CoatingObject() [2/2]	51
7.5.3	Методы	51
7.5.3.1	GetSprite()	51
7.5.3.2	GetResistance()	51
7.5.3.3	HasDirection()	51
7.5.3.4	IsDirectionAvailable()	52
7.5.3.5	SetDirectionAvailability()	52
7.5.3.6	SetSprite_()	52
7.5.3.7	SetX_()	53
7.5.3.8	SetY_()	53
7.6	Класс <code>rtm::CoatingUnion</code>	53
7.6.1	Подробное описание	55
7.6.2	Конструктор(ы)	55
7.6.2.1	CoatingUnion()	55
7.6.3	Методы	55
7.6.3.1	GetType()	56
7.6.3.2	GetWidth()	56
7.6.3.3	GetHeight()	56
7.6.3.4	GetCoatingObject()	56
7.6.3.5	GetLength()	57
7.6.3.6	IsCorrectColumn()	57
7.6.3.7	IsCorrectRow()	57
7.6.3.8	ShowSprites()	57

7.6.3.9	ReleaseSprites()	58
7.6.3.10	GetColumn_()	58
7.6.3.11	GetRow_()	58
7.7	Класс rtm::ControlUnit	59
7.7.1	Подробное описание	60
7.7.2	Конструктор(ы)	60
7.7.2.1	ControlUnit() [1/2]	60
7.7.2.2	ControlUnit() [2/2]	61
7.7.3	Методы	61
7.7.3.1	Update()	61
7.7.3.2	operator bool()	61
7.7.3.3	GetSignal()	62
7.7.3.4	ShowSprites()	62
7.7.3.5	ReleaseSprites()	62
7.7.3.6	UpdateSignal_()	62
7.7.3.7	SetState_()	63
7.8	Класс rtm::CrossroadObject	63
7.8.1	Подробное описание	64
7.8.2	Конструктор(ы)	64
7.8.2.1	CrossroadObject() [1/2]	65
7.8.2.2	CrossroadObject() [2/2]	65
7.8.3	Методы	65
7.8.3.1	CrossroadMatrix()	65
7.8.3.2	TCrossroadMatrix()	66
7.8.3.3	GetNullDirection()	66
7.8.3.4	GetControlUnit()	67
7.8.3.5	ShowSprites()	67
7.8.3.6	ReleaseSprites()	67
7.9	Класс rtm::DrivewayObject	68
7.9.1	Подробное описание	69

7.9.2	Конструктор(ы)	69
7.9.2.1	DrivewayObject()	69
7.9.3	Методы	70
7.9.3.1	DrivewayMatrix()	70
7.9.3.2	GetLength()	70
7.9.3.3	GetLinesCount()	71
7.9.3.4	isRightLine() [1/2]	71
7.9.3.5	isRightLine() [2/2]	71
7.9.3.6	isLeftLine() [1/2]	72
7.9.3.7	isLeftLine() [2/2]	72
7.9.3.8	CountLength_()	72
7.9.3.9	CountLines_()	73
7.10	Класс rtm::DynamicObject	73
7.10.1	Подробное описание	75
7.10.2	Конструктор(ы)	75
7.10.2.1	DynamicObject() [1/2]	75
7.10.2.2	DynamicObject() [2/2]	76
7.10.3	Методы	76
7.10.3.1	GetSpeed()	76
7.10.3.2	GetLastDelta()	77
7.10.3.3	HasCollision()	77
7.10.3.4	Update()	77
7.10.3.5	IsNearOthers()	77
7.10.3.6	SetSpeed_()	78
7.10.3.7	SetCollisionFlag_()	78
7.10.3.8	IsBeholding_()	78
7.10.3.9	IsIntersecting_()	79
7.10.3.10	IsNear_()	79
7.10.4	Документация по друзьям класса и функциям, относящимся к классу	80
7.10.4.1	CheckCollisions	80



7.11	Класс <code>rtm::MapObject</code> . . . . .	80
7.11.1	Подробное описание . . . . .	81
7.11.2	Конструктор(ы) . . . . .	81
7.11.2.1	<code>MapObject()</code> [1/2] . . . . .	81
7.11.2.2	<code>MapObject()</code> [2/2] . . . . .	81
7.12	Класс <code>rtm::PuddleCoating</code> . . . . .	83
7.12.1	Подробное описание . . . . .	84
7.12.2	Конструктор(ы) . . . . .	84
7.12.2.1	<code>PuddleCoating()</code> [1/3] . . . . .	84
7.12.2.2	<code>PuddleCoating()</code> [2/3] . . . . .	84
7.12.2.3	<code>PuddleCoating()</code> [3/3] . . . . .	85
7.13	Класс <code>rtm::RoadCoating</code> . . . . .	85
7.13.1	Подробное описание . . . . .	86
7.13.2	Конструктор(ы) . . . . .	86
7.13.2.1	<code>RoadCoating()</code> [1/3] . . . . .	86
7.13.2.2	<code>RoadCoating()</code> [2/3] . . . . .	87
7.13.2.3	<code>RoadCoating()</code> [3/3] . . . . .	87
7.13.3	Методы . . . . .	87
7.13.3.1	<code>GetClassResistance_()</code> . . . . .	88
7.13.3.2	<code>GetClassDirections_()</code> . . . . .	88
7.14	Структура <code>rtm::SpawnType</code> . . . . .	88
7.14.1	Подробное описание . . . . .	89
7.15	Класс <code>rtm::StaticObject</code> . . . . .	89
7.15.1	Подробное описание . . . . .	90
7.15.2	Конструктор(ы) . . . . .	90
7.15.2.1	<code>StaticObject()</code> [1/2] . . . . .	90
7.15.2.2	<code>StaticObject()</code> [2/2] . . . . .	90
7.16	Класс <code>rtm::TurnObject</code> . . . . .	91
7.16.1	Подробное описание . . . . .	92
7.16.2	Конструктор(ы) . . . . .	92

7.16.2.1	TurnObject()	92
7.16.3	Методы	92
7.16.3.1	RightTurnMatrix()	92
7.16.3.2	LeftTurnMatrix()	93
7.16.3.3	IsRight()	93
7.16.3.4	GetAngle()	94
7.17	Класс rtm::VehicleObject	94
7.17.1	Подробное описание	97
7.17.2	Конструктор(ы)	97
7.17.2.1	VehicleObject() [1/2]	97
7.17.2.2	VehicleObject() [2/2]	98
7.17.3	Методы	98
7.17.3.1	Update()	98
7.17.3.2	MoveForward_()	99
7.17.3.3	Stop_()	99
7.17.3.4	Rotate_()	99
7.17.3.5	ChangeLine_()	100
7.17.3.6	IsMovement_()	100
7.17.3.7	IsRotation_()	100
7.17.3.8	IsLineChanging_()	100
7.17.3.9	IsBraking_()	101
7.17.3.10	GetMaxSpeed_()	101
7.17.3.11	GetFinalSpeed_()	101
7.17.3.12	SetFinalSpeed_()	101
7.17.3.13	SetBrakingFactor_()	102
7.17.3.14	StopAtDistance_()	102
7.17.3.15	CheckForwardCoating_()	102
7.17.3.16	CheckForwardCoatingUnion_()	103
7.17.3.17	CheckForwardArea_()	103
7.17.3.18	CheckMovingArea_()	103

7.17.3.19 CheckTurnArea_()	104
7.17.3.20 CheckRotationArea_()	104
7.17.3.21 CheckCrossroadArea_()	105
7.17.3.22 CheckLineChangingArea_()	105
7.17.3.23 BeforeMoving_()	105
7.17.3.24 AfterMoving_()	106
7.17.3.25 MovementStart_()	106
7.17.3.26 MovementTick_()	106
7.17.3.27 MovementEnd_()	107
7.17.3.28 RotationStart_()	107
7.17.3.29 RotationTick_()	107
7.17.3.30 RotationEnd_()	108
7.17.3.31 LineChangingStart()	108
7.17.3.32 LineChangingTick_()	109
7.17.3.33 LineChangingEnd_()	109
7.17.3.34 LineChanging_()	109
7.17.3.35 Rotation_()	110
7.17.3.36 Movement_()	110
7.17.3.37 SpeedChanging_()	110
7.17.3.38 SmoothBrakingCounter()	110
7.18 Класс rtm::WorldController	111
7.18.1 Подробное описание	114
7.18.2 Конструктор(ы)	114
7.18.2.1 WorldController() [1/3]	114
7.18.2.2 WorldController() [2/3]	114
7.18.2.3 WorldController() [3/3]	115
7.18.3 Методы	115
7.18.3.1 Update()	115
7.18.3.2 GetLayer()	115
7.18.3.3 GetColumnsCount()	116

7.18.3.4	GetRowCount()	116
7.18.3.5	GetDeltaTime()	116
7.18.3.6	GetTimeFactor()	116
7.18.3.7	GetCoatingObject()	117
7.18.3.8	GetCoatingUnion()	117
7.18.3.9	GetStaticObject()	117
7.18.3.10	GetDynamicObjects()	117
7.18.3.11	IsPause()	118
7.18.3.12	IsCorrectColumn()	118
7.18.3.13	IsCorrectRow()	118
7.18.3.14	IsAllowableColumn()	119
7.18.3.15	IsAllowableRow()	119
7.18.3.16	IsVisibleColumn()	119
7.18.3.17	IsVisibleRow()	120
7.18.3.18	SetTimeFactor()	120
7.18.3.19	LoadMap() [1/2]	120
7.18.3.20	LoadMap() [2/2]	121
7.18.3.21	IsEmpty_()	121
7.18.3.22	GenerateObject_()	121
7.18.3.23	AddCoatingUnion_()	122
7.18.3.24	AddDriveway_()	122
7.18.3.25	AddCrossroad_()	123
7.18.3.26	AddTCrossroad_()	123
7.18.3.27	AddLeftTurt_()	124
7.18.3.28	AddRightTurt_()	124
7.18.3.29	AddControlUnit_()	125
7.18.3.30	AddStaticObject_()	125
7.18.3.31	AddBuilding_()	125
7.18.3.32	AddBush_()	126
7.18.3.33	AddDynamicObject_()	126

7.18.3.34	AddCar_()	127
7.18.3.35	GetVectorColumn_()	127
7.18.3.36	GetVectorRow_()	128
7.18.3.37	GetRealColumn_()	128
7.18.3.38	GetRealRow_()	128
7.19	Класс rtm::WorldObject	129
7.19.1	Подробное описание	131
7.19.2	Конструктор(ы)	131
7.19.2.1	WorldObject() [1/2]	131
7.19.2.2	WorldObject() [2/2]	132
7.19.3	Методы	132
7.19.3.1	GetSprite()	132
7.19.3.2	GetX_()	132
7.19.3.3	GetY_()	133
7.19.3.4	GetAngle()	133
7.19.3.5	GetWidth()	133
7.19.3.6	GetHeight()	133
7.19.3.7	SetSprite_()	133
7.19.3.8	SetX_()	134
7.19.3.9	SetY_()	134
7.19.3.10	SetAngle_()	134
7.19.3.11	SetWidth_()	135
7.19.3.12	SetHeight_()	135
7.19.3.13	SetSpriteX_()	135
7.19.3.14	SetSpriteY_()	135
7.19.3.15	SetSpriteAngle_()	136
7.19.3.16	SetSpriteWidth_()	136
7.19.3.17	SetSpriteHeight_()	136
7.20	Класс rtm::WorldScene	137
7.20.1	Подробное описание	139
7.20.2	Методы	139
7.20.2.1	Create()	139
7.20.2.2	init()	140
7.20.2.3	update()	140
7.20.2.4	GetMainLayer()	140
7.20.2.5	SetBackground() [1/2]	140
7.20.2.6	SetBackground() [2/2]	141
7.20.2.7	GetMap_()	141



# 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 . . . . .	35
rtm::CoatingObject . . . . .	48
rtm::PuddleCoating . . . . .	83
rtm::RoadCoating . . . . .	85
rtm::CoatingUnion . . . . .	53
rtm::CrossroadObject . . . . .	63
rtm::DrivewayObject . . . . .	68
rtm::TurnObject . . . . .	91
rtm::ControlUnit . . . . .	59
Scene	
rtm::WorldScene . . . . .	137
rtm::SpawnType . . . . .	88
rtm::WorldController . . . . .	111
rtm::WorldObject . . . . .	129
rtm::DynamicObject . . . . .	73
rtm::VehicleObject . . . . .	94
rtm::CarObject . . . . .	41
rtm::StaticObject . . . . .	89
rtm::MapObject . . . . .	80
rtm::BuildingObject . . . . .	36
rtm::BushObject . . . . .	38



## Глава 5

# Алфавитный указатель классов

### 5.1 Классы

Классы с их кратким описанием.

<a href="#">AppDelegate</a>	Приложение, основанное на Cocos2d . . . . .	35
<a href="#">rtm::BuildingObject</a>	Класс, описывающий строения (здания) . . . . .	36
<a href="#">rtm::BushObject</a>	Класс, описывающий кусты . . . . .	38
<a href="#">rtm::CarObject</a>	Класс, описывающий машины . . . . .	41
<a href="#">rtm::CoatingObject</a>	Класс покрытия секции карты . . . . .	48
<a href="#">rtm::CoatingUnion</a>	Класс объединения покрытий . . . . .	53
<a href="#">rtm::ControlUnit</a>	Класс управляющего блока (светофор) . . . . .	59
<a href="#">rtm::CrossroadObject</a>	Класс пересечения дорог . . . . .	63
<a href="#">rtm::DrivewayObject</a>	Класс прямой дороги . . . . .	68
<a href="#">rtm::DynamicObject</a>	Класс динамического объекта (который движется, обновляется) . . . . .	73
<a href="#">rtm::MapObject</a>	Класс статического объекта карты . . . . .	80
<a href="#">rtm::PuddleCoating</a>	Класс, описывающий лужи . . . . .	83
<a href="#">rtm::RoadCoating</a>	Класс, описывающий дороги . . . . .	85
<a href="#">rtm::SpawnType</a>	Структура, описывающая параметры точки генерации объектов . . . . .	88
<a href="#">rtm::StaticObject</a>	Класс статического объекта (который не обновляется) . . . . .	89
<a href="#">rtm::TurnObject</a>	Класс поворота дороги . . . . .	91
<a href="#">rtm::VehicleObject</a>	Класс транспорта (динамического объекта карты) . . . . .	94
<a href="#">rtm::WorldController</a>	Класс контроллера карты, связующее звено всех объектов . . . . .	111

<a href="#">rtm::WorldObject</a>	
Класс объекта мира (родитель всех условно объемных объектов) . . . . .	<a href="#">129</a>
<a href="#">rtm::WorldScene</a>	
Класс главной сцены, на которой всё и происходит (для отрисовки) . . . . .	<a href="#">137</a>



## Глава 6

# Пространства имен

### 6.1 Пространство имен rtm

Пространство имен для проекта RTM.

#### Классы

- class [BuildingObject](#)  
Класс, описывающий строения (здания)
- class [BushObject](#)  
Класс, описывающий кусты
- class [CarObject](#)  
Класс, описывающий машины
- class [CoatingObject](#)  
Класс покрытия секции карты
- class [CoatingUnion](#)  
Класс объединения покрытий
- class [ControlUnit](#)  
Класс управляющего блока (светофор)
- class [CrossroadObject](#)  
Класс пересечения дорог
- class [DrivewayObject](#)  
Класс прямой дороги
- class [DynamicObject](#)  
Класс динамического объекта (который движается, обновляется)
- class [MapObject](#)  
Класс статического объекта карты
- class [PuddleCoating](#)  
Класс, описывающий лужи
- 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 >  
Массив возможных направлений движений по кусочку объекта
- using [LinesCounts](#) = std::array< size\_t, 4 >  
Массив количества полос в каждом направлении для перекрестков
- using [DirectionSignals](#) = std::array< [SignalType](#), 4 >  
Массив сигналов из одного направления в каждое
- using [CrossroadSignals](#) = std::array< [DirectionSignals](#), 4 >

- Массив сигналов для всех направлений перекрестка  
 using `SignalSprites` = `std::array< cocos2d::Sprite *, 5 >`
- Массив всех текстур сигналов из одного направления в одно  
 using `SignalsSprites` = `std::array< SignalSprites, 3 >`
- Массив всех текстур сигналов из одного направления в каждое (вперед, влево, вправо)  
 using `DirectionsSignalSprites` = `std::array< SignalsSprites, 4 >`
- Массив всех текстур сигналов для перекрестка

## 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, `IceAsphaltCoating` = 1 }
- Типы покрытий
- enum `RoadType` {  
`RoadTypeNo0` = 0, `RoadTypeNo1`, `RoadTypeNo2`, `RoadTypeNo3`,  
`RoadTypeNo4`, `RoadTypeNo5`, `RoadTypeNo6`, `RoadTypeNo7`,  
`RoadTypeNo8`, `RoadTypeNo9`, `RoadTypeNo10`, `RoadTypeNo11`,  
`RoadTypeNo12`, `RoadTypeNo13`, `RoadTypeNo14`, `RoadTypeNo15`,  
`RoadTypeNo16`, `RoadTypeNo17` }
- Типы дорог
- enum `SignalFileId` { `ForwardSignalId` = 1, `LeftwardSignalId` = 6, `RightwardSignalId` = 11 }
- Индексы, начиная с которых начинаются текстуры сигнала определенного типа

## Функции

- void `CheckCollisions` (`WorldController *const world`)  
 Функция для вычисления столкновений в мире
- std::string `GetFilename` (std::string const &mask, size\_t number)  
 Функция для получения пути к файлу по маске
- `AngleType SumAngleTypes` (`AngleType a`, `AngleType b`)

Функция для суммирования двух угловых типов

- float [CountDeceleration](#) (float maxSpeed)

Функция для подсчёта рекомендуемого коэффициента замедления транспорта

Функции для работы с параметрами положения объектов

- bool [SameCoordinates](#) (float a, float b, float delta=[COORD\\_DELTA](#))  
Функция для сравнения двух координат с определенной точностью
- float [RoundCoordinate](#) (float coordinate, float delta=[COORD\\_DELTA](#))  
Функция пытается округлить координаты до центра клетки
- float [RoundToCenter](#) (float coordinate)  
Функция для округления координаты до центра клетки
- bool [InCenter](#) (float coordinate, float delta=[COORD\\_DELTA](#))  
Функция для проверки координаты на центральность
- float [DistanceToNextCenter](#) (float x, float y, float angle)  
Функция для нахождения расстояния до следующего центра клетки по ходу движения
- bool [CenterIsCrossed](#) (float x, float y, float angle, float lastDelta)  
Функция проверяет, пересек ли объект центр клетки (центральную линию, перпендикулярную направлению движения)
- bool [SameAngles](#) (float a, float b, float delta=[ANGLE\\_DELTA](#))  
Функция для сравнения двух углов
- float [RoundAngle](#) (float angle, float delta=[ANGLE\\_DELTA](#))  
Функция пытается округлить угол до одно из главных направлений (период  $\pi/4$ , т.е. 0,  $\pi/4$ ,  $\pi/2$ , ...)
- float [NormalizeAngle](#) (float angle)  
Функция для нормализации угла до диапазона  $[-\pi/2; \pi/2]$

Конверторы схожих типов

- int [PixelToCell](#) (float coordinate)  
Функция для конвертации координаты в номер ячейки
- float [CellToPixel](#) (int cellNumber)  
Функция для конвертации номера ячейки в координату центра
- [AngleType](#) [AngleToAngleType](#) (float angle)  
Функция для конвертации угла в угловой тип
- [DirectionType](#) [AngleToDirection](#) (float angle)  
Функция для конвертации угла в направление
- float [AngleTypeToAngle](#) ([AngleType](#) angle)  
Функция для конвертации углового типа в угол
- [DirectionType](#) [AngleTypeToDirection](#) ([AngleType](#) angle)  
Функция для конвертации углового типа в направление
- float [DirectionToAngle](#) ([DirectionType](#) direction)  
Функция для конвертации направления в угол
- [AngleType](#) [DirectionToAngleType](#) ([DirectionType](#) direction)  
Функция для конвертации направления в угловой тип

Переменные

- `std::array< size_t, 2 > const` [COATING\\_INDEXES](#)  
Индексы, начиная с которых начинаются текстуры покрытий определенного типа

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 }`  
Максимальное расстояние до объектов, которые недалеко

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_LAYER_Z_ORDER` { -1 }  
Номер слоя для слоя фона
- int const `MAIN_LAYER_Z_ORDER` { 0 }  
Номер слоя для главного слоя (на нем все объекты)
- int const `COATING_OBJECT_Z_ORDER` { -2 }  
Номер слоя для покрытий (дорог)
- int const `SIGNAL_Z_ORDER` { -1 }  
Номер слоя для стрелок светофора
- int const `VEHICLE_OBJECT_Z_ORDER` { 0 }  
Номер слоя для транспорта
- int const `MAP_OBJECT_Z_ORDER` { 1 }  
Номер слоя для статичных объектов карты

Область видимости при движении вперед

- float const `VIEW_RADIUS` { 60.f }  
Радиус
- float const `VIEW_ANGLE` { 25.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` { 60.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` { 58.f }  
Радиус
- float const `CROSSROAD_VIEW_ANGLE` { 57.5f \* `DEG_RAD` }  
Ширина угла в каждую сторону
- float const `CROSSROAD_VIEW_ANGLE_SHIFT` { -17.5f \* `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 }
- float const `LINE_CHANGING_VIEW_ANGLE_SHIFT` { 20.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
- `SignalSprites` 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/road/RoadNo%No%.png" }  
Маска файлов текстур дорог
- std::string const `PUDDLE_FILENAME_MASK` { "res/coating/puddle/PuddleNo%No%.png" }  
Маска файлов текстур грязи
- std::string const `SIGNAL_FILENAME_MASK` { "res/signal/SignalNo%No%.png" }  
Маска файлов текстур сигналов
- std::string const `BUILDING_FILENAME_MASK` { "res/static/building/BuildingNo%No%.png" }  
Маска файлов текстур зданий
- std::string const `BUSH_FILENAME_MASK` { "res/static/bush/BushNo%No%.png" }  
Маска файлов текстур кустов
- std::string const `CAR_FILENAME_MASK` { "res/dynamic/vehicle/CarNo%No%.png" }  
Маска файлов текстур машин

## Параметры дорог

- std::array< float, 2 > const `ROADS_RESISTANCES`  
Массив коэффициентов трения для каждого типа объекта
- std::array< `Directions`, 18 > const `ROADS DIRECTIONS`  
Массив возможных направлений для каждой типа кучоска дороги

## Параметры машин

- std::array< float, 6 > const `CARS_MAX_SPEEDS`  
Массив максимальных скоростей для машин
- std::array< float, 6 > const `CARS_ACCELERATIONS`  
Массив ускорений для машин

### 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`

Типы покрытий

Элементы перечислений

AsphaltCoating	Асфальтовое покрытие
IceAsphaltCoating	Асфальтовое покрытие со льдом

## 6.1.3.8 RoadType

enum `rtm::RoadType`

Типы дорог

Элементы перечислений

RoadTypeNo0	Однополосная прямая
RoadTypeNo1	Однополосная левая прямая
RoadTypeNo2	Однополосная средняя прямая
RoadTypeNo3	Перекресток 1 на 1.
RoadTypeNo4	Угол перекрестка 1 на N.
RoadTypeNo5	Угол перекрестка N на N.
RoadTypeNo6	Центральная часть перекрестка
RoadTypeNo7	T-образный перекресток 1 на 1.
RoadTypeNo8	Левый угол T-образного перекрестка 1 на N.
RoadTypeNo9	Заблокированный край T-образного перекрестка N на N.
RoadTypeNo10	Правый угол T-образного перекрестка 1 на N.
RoadTypeNo11	Расширение дороги
RoadTypeNo12	Сужение дороги
RoadTypeNo13	Однополосный поворот
RoadTypeNo14	Внешний ряд поворота
RoadTypeNo15	Средний ряд поворота
RoadTypeNo16	Внутренний ряд поворота
RoadTypeNo17	Обочина поворота (угла перекрестка)

## 6.1.3.9 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 GetFilename()

```
std::string rtm::GetFilename (
    std::string const & mask,
    size_t number )
```

Функция для получения пути к файлу по маске

Аргументы

mask	маска названия файла
number	номер объекта

Возвращает

путь к файлу

#### 6.1.4.20 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 ( $a + b$ )

#### 6.1.4.21 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 COATING\_INDEXES

```
std::array<size_t, 2> const rtm::COATING_INDEXES
```

Инициализатор

```
= {
    1
    , 21
}
```

Индексы, начиная с которых начинаются текстуры покрытий определенного типа

#### 6.1.5.5 ROADS\_RESISTANCES

```
std::array<float, 2> const rtm::ROADS_RESISTANCES
```

Инициализатор

```
= {
    1.f
    , 0.8f
}
```

Массив коэффициентов трения для каждого типа объекта

См. также

[CoatingType](#)

#### 6.1.5.6 ROADS DIRECTIONS

```
std::array<Directions, 18> const rtm::ROADS DIRECTIONS
```

Инициализатор

```
= {
    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, false, false, false, false, false, false, false }
}
```

Массив возможных направлений для каждой типа кучоска дороги

См. также

[RoadCoating](#)

#### 6.1.5.7 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](#)

#### 6.1.5.8 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

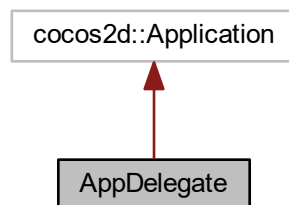
# Классы

### 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 ()`  
Функция для инициализации Director'a и Scene'ы
- `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:

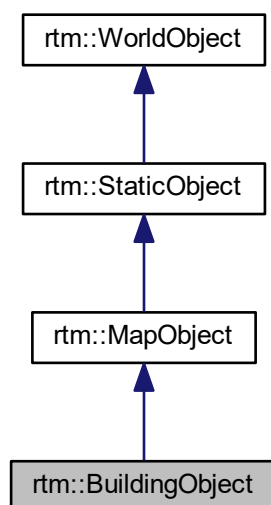
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/AppDelegate.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/AppDelegate.cpp

## 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)  
Конструктор с использованием уже готового спрайта
- [BuildingObject](#) (std::string const &filename, int column, int row, float angle)  
Конструктор из файла
- [BuildingObject](#) (size\_t type, int column, int row, float angle)  
Конструктор стандартного строения
- virtual [~BuildingObject](#) ()=default  
Деструктор по умолчанию

## 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

Объявления и описания членов классов находятся в файлах:

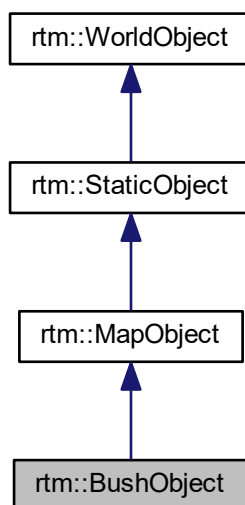
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BuildingObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BuildingObject.cpp

## 7.3 Класс rtm::BushObject

Класс, описывающий кусты

```
#include <BushObject.h>
```

Граф наследования: `rtm::BushObject`:



#### Открытые члены

- `BushObject` ()  
Конструктор по умолчанию
- `BushObject` (`cocos2d::Sprite *const sprite`, `int column`, `int row`, `float angle`)  
Конструктор с использованием уже готового спрайта
- `BushObject` (`std::string const &filename`, `int column`, `int row`, `float angle`)  
Конструктор из файла
- `BushObject` (`size_t type`, `int column`, `int row`, `float angle`)  
Конструктор стандартного куста
- `virtual ~BushObject` ()=default  
Деструктор по умолчанию

#### Дополнительные унаследованные члены

##### 7.3.1 Подробное описание

Класс, описывающий кусты

##### 7.3.2 Конструктор(ы)

## 7.3.2.1 BushObject() [1/3]

```
rtm::BushObject::BushObject (
    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 куст
row	строка, в которой необходимо отрисовать куст
angle	угол поворота куста

## 7.3.2.2 BushObject() [2/3]

```
rtm::BushObject::BushObject (
    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 куст
row	строка, в которой необходимо отрисовать куст
angle	угол поворота куста

## 7.3.2.3 BushObject() [3/3]

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

Конструктор стандартного куста

## Аргументы

type	стандартный тип куста
column	колонка, в которой необходимо отрисовать куст
row	строка, в которой необходимо отрисовать куст
angle	угол поворота куста

Объявления и описания членов классов находятся в файлах:

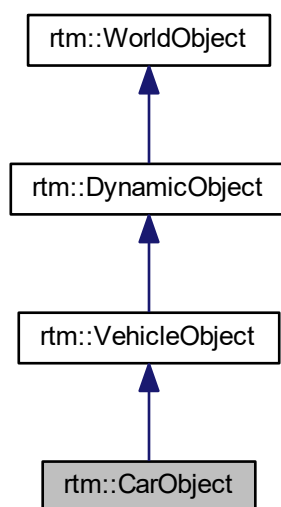
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BushObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/BushObject.cpp

## 7.4 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)  
Конструктор с использованием уже готового спрайта

- `CarObject` (`std::string const &filename`, `int column`, `int row`, `float angle`, `float maxSpeed`, `float acceleration`)  
Конструктор из файла
- `CarObject` (`size_t type`, `int column`, `int row`, `float angle`)  
Конструктор стандартной машины
- `virtual ~CarObject ()=default`  
Деструктор по умолчанию

## Защищенные члены

- `virtual bool MovementStart_ (WorldController *const world) override`  
Функция, которая просто пропускает выполнение родителя
- `virtual bool MovementTick_ (WorldController *const world) override`  
Функция для вычисления скорости
- `virtual bool MovementEnd_ (WorldController *const world) override`  
Функция обнуляет финальную скорость
- `virtual bool LineChangingStart (WorldController *const world) override`  
Функция, описывающая движение перед перестроением

## Закрытые члены

- `void SetDesiredSpeed_ (float speed)`  
Функция для установки желаемой скорости
- `void ResetDesiredSpeed_ ()`  
Функция для сброса желаемой скорости
- `void CheckCoatingAhead_ (WorldController *const world)`  
Функция для проверки объекта (повороты и т.д.)
- `void CheckCoatingUnionAhead_ (WorldController *const world)`  
Функция для проверки объединения покрытий (заранее тормозим перед светофорами и т.д.)
- `void CheckRoadAhead_ (WorldController *const world)`  
Функция для проверки дороги спереди (принятие решений)

## Закрытые статические члены

- `static float GetClassMaxSpeed_ (size_t id)`  
Функция для получения максимальной скорости стандартной машины по номеру
- `static float GetClassAcceleration_ (size_t id)`  
Функция для получения ускорения стандартной машины по номеру



## 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)

### 7.4.1 Подробное описание

Класс, описывающий машины

### 7.4.2 Конструктор(ы)

#### 7.4.2.1 `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

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

#### 7.4.2.2 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

#### 7.4.2.3 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

### 7.4.3 Методы

7.4.3.1 `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](#).

7.4.3.2 `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](#).

7.4.3.3 `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](#).

7.4.3.4 `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](#).

#### 7.4.3.5 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
-------	---

#### 7.4.3.6 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
-------	---

#### 7.4.3.7 CheckRoadAhead\_()

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

Функция для проверки дороги спереди (принятие решений)

Аргументы

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

#### 7.4.3.8 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

#### 7.4.3.9 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:

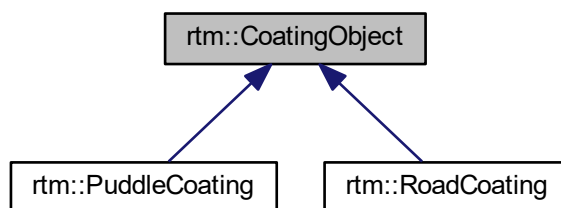
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CarObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CarObject.cpp

## 7.5 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`)  
Конструктор с использованием уже готового спрайта
- `CoatingObject` (`std::string const &filename`, `int column`, `int row`, `AngleType angle`, `float resistance`, `Directions directions`)  
Конструктор из файла
- `virtual ~CoatingObject` ()=default  
Деструктор по умолчанию
- `cocos2d::Sprite * GetSprite` () const  
Функция для получения спрайта
- `float GetResistance` () const  
Функция для получения коэффициента сопротивления на покрытии
- `bool HasDirection` (`AngleType angle`) const  
Функция для проверки существования направления
- `bool IsDirectionAvailable` (`AngleType angle`) const  
Функция для проверки разрешенности направления
- `void SetDirectionAvailability` (`AngleType angle`, `bool status`)  
Функция установки разрешенности направления

### Защищенные члены

- `void SetSprite_` (`cocos2d::Sprite *const sprite`)  
Функция для установки спрайта

### Закрытые члены

- `void SetX_` (`float x`)  
Функция для установки абсциссы
- `void SetY_` (`float y`)  
Функция для установки ординаты

## 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

### 7.5.1 Подробное описание

Класс покрытия секции карты

### 7.5.2 Конструктор(ы)

#### 7.5.2.1 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



## 7.5.2.2 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

## 7.5.3 Методы

## 7.5.3.1 GetSprite()

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

Функция для получения спрайта

Возвращает

указатель на спрайт

## 7.5.3.2 GetResistance()

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

Функция для получения коэффициента сопротивления на покрытии

Возвращает

сопротивление

## 7.5.3.3 HasDirection()

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

Функция для проверки существования направления

Аргументы

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

Возвращает

true, если доступно (существует), иначе false

#### 7.5.3.4 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

#### 7.5.3.5 SetDirectionAvailability()

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

Функция установки разрешенности направления

Аргументы

angle	направление
status	разрешено ли ехать

#### 7.5.3.6 SetSprite\_()

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

Функция для установки спрайта

Аргументы

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

#### 7.5.3.7 SetX\_()

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

Функция для установки абсциссы

Аргументы

x	абсцисса
---	----------

#### 7.5.3.8 SetY\_()

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

Функция для установки ординаты

Аргументы

y	ордината
---	----------

Объявления и описания членов классов находятся в файлах:

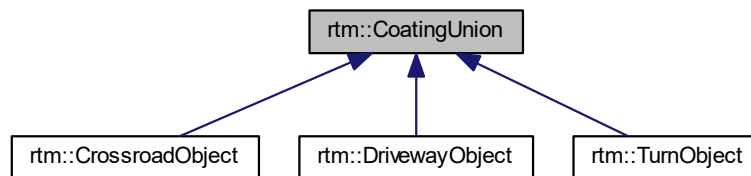
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CoatingObject.cpp

## 7.6 Класс rtm::CoatingUnion

Класс объединения покрытий

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

Граф наследования: `rtm::CoatingUnion`:



## Открытые члены

- [CoatingUnion](#) ()  
Конструктор по умолчанию
- [CoatingUnion](#) ([CoatingUnionType](#) type, int column, int row, [CoatingMatrix](#) &&objects)  
Конструктор по матрице покрытий
- `virtual ~CoatingUnion ()=default`  
Деструктор по умолчанию
- [CoatingUnionType GetType](#) () const  
Функция для получения типа объединения
- `size_t GetWidth` () const  
Функция для получения ширины объединения
- `size_t GetHeight` () const  
Функция для получения высоты объединения
- [CoatingUnique](#) const & [GetCoatingObject](#) (int column, int row) const  
Функция для получения объекта
- `virtual float GetLength` () const  
Функция для получения длины (количества покрытий)
- `bool IsCorrectColumn` (int column) const  
Функция для проверки корректности колонки в данном объединении
- `bool IsCorrectRow` (int row) const  
Функция для проверки корректности строки в данном объединении
- `virtual void ShowSprites` (cocos2d::Layer \*const layer)  
Функция для добавления спрайтов на сцену
- `virtual void ReleaseSprites` (cocos2d::Layer \*const layer)  
Функция для удаления спрайтов со сцены

## Защищенные члены

- `int GetColumn_` () const  
Функция для получения левой колонки объединения
- `int GetRow_` () const  
Функция для получения нижней строки объединения

## 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

### 7.6.1 Подробное описание

#### Класс объединения покрытий

См. также

[CoatingObject](#)

### 7.6.2 Конструктор(ы)

#### 7.6.2.1 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

### 7.6.3 Методы

#### 7.6.3.1 GetType()

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

Функция для получения типа объединения

Возвращает

тип объединения

#### 7.6.3.2 GetWidth()

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

Функция для получения ширины объединения

Возвращает

ширина объединения

#### 7.6.3.3 GetHeight()

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

Функция для получения высоты объединения

Возвращает

высота объединения

#### 7.6.3.4 GetCoatingObject()

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

Функция для получения объекта

Аргументы

column	колонка (относительно всей карты), в которой находится объект
row	строка (относительно всей карты), в которой находится объект

Возвращает

умный указатель на объект

#### 7.6.3.5 `GetLength()`

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

Функция для получения длины (количества покрытий)

Возвращает

длина

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

#### 7.6.3.6 `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`

#### 7.6.3.7 `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`

#### 7.6.3.8 `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](#).

## 7.6.3.9 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](#).

## 7.6.3.10 GetColumn\_()

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

Функция для получения левой колонки объединения

Возвращает

левая колонка объединения

## 7.6.3.11 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



## 7.7 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)  
Конструктор управляющего блока перекрестком
- [ControlUnit](#) (size\_t type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection)  
Конструктор управляющего блока перекрестком
- virtual [~ControlUnit](#) ()=default  
Деструктор по умолчанию
- void [Update](#) ([WorldController](#) \*const world)  
Функция обновления
- [operator bool](#) () const  
Оператор преобразования в логический тип
- [SignalType](#) [GetSignal](#) ([DirectionType](#) from, [DirectionType](#) to) const  
Функция для получения сигнала (статуса направления)
- void [ShowSprites](#) (cocos2d::Layer \*const layer)  
Функция для добавления спрайтов на сцену
- void [ReleaseSprites](#) (cocos2d::Layer \*const layer)  
Функция для удаления спрайтов со сцены

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)  
Функция для безопасной установки сигнала (если направление не закрыто)
- void [IncState](#) \_ ()  
Функция для инкремента номера состояния
- void [SetState](#) \_ (size\_t state)  
Функция для установки номера состояния
- void [ResetState](#) \_ ()  
Функция для сброса номера состояния

## 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)

### 7.7.1 Подробное описание

Класс управляющего блока (светофор)

### 7.7.2 Конструктор(ы)

#### 7.7.2.1 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

## 7.7.2.2 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

## 7.7.3 Методы

## 7.7.3.1 Update()

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

Функция обновления

Аргументы

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

## 7.7.3.2 operator bool()

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

Оператор преобразования в логический тип

Возвращает

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

### 7.7.3.3 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

### 7.7.3.4 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
-------	---

### 7.7.3.5 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
-------	--

### 7.7.3.6 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

#### 7.7.3.7 SetState\_()

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

Функция для установки номера состояния

Аргументы

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

Объявления и описания членов классов находятся в файлах:

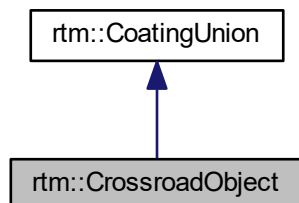
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/ControlUnit.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/ControlUnit.cpp

## 7.8 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)  
Конструктор для обычного перекрестка
- [CrossroadObject](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection, size\_t controlUnitType=0)  
Конструктор для Т-образного перекрестка
- virtual [~CrossroadObject](#) ()=default  
Деструктор по умолчанию
- [AngleType](#) [GetNullDirection](#) () const  
Функция для получения стороны, в направлении которой нельзя двигаться
- [ControlUnitShared](#) [GetControlUnit](#) () const  
Функция для получения управляющего блока, привязанного к данному объекту
- virtual void [ShowSprites](#) (cocos2d::Layer \*const layer) override  
Функция для добавления спрайтов на сцену
- virtual void [ReleaseSprites](#) (cocos2d::Layer \*const layer) override  
Функция для удаления спрайтов со сцены

## Открытые статические члены

- static [CoatingMatrix](#) [CrossroadMatrix](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts)  
Функция для получения матрицы покрытий перекрестка
- static [CoatingMatrix](#) [TCrossroadMatrix](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection)  
Функция для получения матрицы покрытий Т-образного перекрестка

## 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

### 7.8.1 Подробное описание

#### Класс пересечения дорог

### 7.8.2 Конструктор(ы)

## 7.8.2.1 CrossroadObject() [1/2]

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

Конструктор для обычного перекрестка

Аргументы

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

## 7.8.2.2 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

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

## 7.8.3 Методы

## 7.8.3.1 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

#### 7.8.3.2 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

#### 7.8.3.3 GetNullDirection()

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

Функция для получения стороны, в направлении которой нельзя двигаться



Возвращает

угол, соответствующий запрещенной стороне

#### 7.8.3.4 `GetControlUnit()`

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

Функция для получения управляющего блока, привязанного к данному объекту

Возвращает

умный указатель на управляющий блок

#### 7.8.3.5 `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`.

#### 7.8.3.6 `ReleaseSprites()`

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

Функция для удаления спрайтов со сцены

Аргументы

<code>layer</code>	слой, с которого надо удалить спрайты управляющего блока
--------------------	--

Переопределяет метод предка `rtm::CoatingUnion`.

Объявления и описания членов классов находятся в файлах:

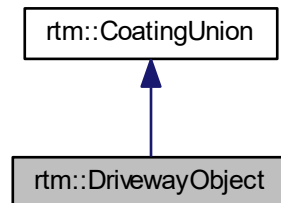
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CrossroadObject.h`
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/CrossroadObject.cpp`

## 7.9 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)  
Конструктор по размерам
- virtual [~DrivewayObject](#) ()=default  
Деструктор по умолчанию
- virtual float [GetLength](#) () const override  
Функция для получения длины объекта (для вычисления кратчайшего пути)
- size\_t [GetLinesCount](#) () const  
Функция для получения количества полос
- bool [isRightLine](#) (int column, int row) const  
Функция для проверки: находится ли объект в правой полосе
- bool [isRightLine](#) (float x, float y) const  
Функция для проверки: находится ли объект в правой полосе
- bool [isLeftLine](#) (int column, int row) const  
Функция для проверки: находится ли объект в левой полосе
- bool [isLeftLine](#) (float x, float y) const  
Функция для проверки: находится ли объект в левой полосе

### Открытые статические члены

- static [CoatingMatrix DrivewayMatrix](#) ([CoatingType](#) type, int column, int row, size\_t width, size\_t height, [AngleType](#) angle)  
Функция для получения матрицы покрытий дороги

## Закрытые статические члены

- static float `CountLength_` (size\_t width, size\_t height, `AngleType` angle)  
Функция для вычисления длины объекта
- static size\_t `CountLines_` (size\_t width, size\_t height, `AngleType` angle)  
Функция для получения количества полос

## Закрытые данные

- `AngleType` `angle_`  
Направление движения
- float `length_`  
Длина объекта (для вычисления кратчайшего пути)
- size\_t `linesCount_`  
Количество полос

## Дополнительные унаследованные члены

### 7.9.1 Подробное описание

#### Класс прямой дороги

### 7.9.2 Конструктор(ы)

#### 7.9.2.1 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

### 7.9.3 Методы

#### 7.9.3.1 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

#### 7.9.3.2 GetLength()

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

Функция для получения длины объекта (для вычисления кратчайшего пути)

Возвращает

длина

Переопределяет метод предка `rtm::CoatingUnion`.

7.9.3.3 `GetLinesCount()`

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

Функция для получения количества полос

Возвращает

количество полос

7.9.3.4 `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`

7.9.3.5 `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`

## 7.9.3.6 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

## 7.9.3.7 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

## 7.9.3.8 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

## 7.9.3.9 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:

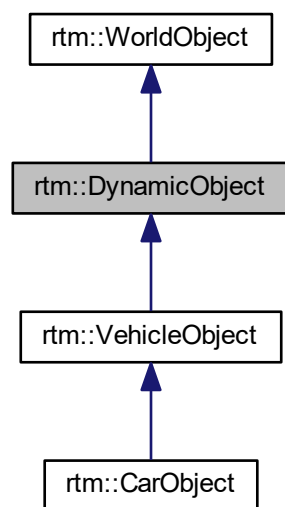
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DrivewayObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DrivewayObject.cpp

## 7.10 Класс rtm::DynamicObject

Класс динамического объекта (который двигается, 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)  
Конструктор с использованием уже готового спрайта
- [DynamicObject](#) (std::string const &filename, float x, float y, float angle, float speed)  
Конструктор из файла
- virtual [~DynamicObject](#) ()=default  
Деструктор по умолчанию
- float [GetSpeed](#) () const  
Функция для получения скорости
- float [GetLastDelta](#) () const  
Функция для получения последнего приращения положения
- bool [HasCollision](#) () const  
Функция для проверки наличия столкновений у данного объекта
- virtual void [Update](#) ([WorldController](#) \*const world)  
Функция обновления
- bool [IsNearOthers](#) ([WorldController](#) \*const world)  
Функция для поиска объектов неподалеку

#### Защищенные члены

- void [SetSpeed\\_](#) (float speed)  
Функция для установки скорости
- void [SetCollisionFlag\\_](#) (bool flag)



Функция для сохранения информации о столкновениях

- `bool IsBeholding_ (WorldObject const *const other, float radius=VIEW_RADIUS, float angle=VIEW_ANGLE, float angleShift=VIEW_ANGLE_SHIFT) const`

Функция для проверки попадания объекта в зону видимости

- `bool IsIntersecting_ (WorldObject const *const other) const`

Функция для проверки наличия столкновения с `other`.

### Закрытые члены

- `bool IsNear_ (WorldObject const *const other) const`

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

### Закрытые данные

- `float speed_`

Текущая скорость

- `float lastDelta_`

Длина последнего смещения

- `bool hasCollision_`

Наличие столкновений у данного объекта

### Друзья

- `void CheckCollisions (WorldController *const world)`

Функция для вычисления столкновений в мире

#### 7.10.1 Подробное описание

Класс динамического объекта (который двигается, обновляется)

#### 7.10.2 Конструктор(ы)

##### 7.10.2.1 `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

## 7.10.2.2 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

## 7.10.3 Методы

## 7.10.3.1 GetSpeed()

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

## Функция для получения скорости

## Возвращает

скорость объекта

7.10.3.2 `GetLastDelta()`

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

Функция для получения последнего приращения положения

Возвращает

длина последнего смещения

7.10.3.3 `HasCollision()`

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

Функция для проверки наличия столкновений у данного объекта

Возвращает

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

7.10.3.4 `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](#).

7.10.3.5 `IsNearOthers()`

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

Функция для поиска объектов неподалеку

## Аргументы

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

## Возвращает

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

## 7.10.3.6 SetSpeed\_()

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

## Функция для установки скорости

## Аргументы

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

## 7.10.3.7 SetCollisionFlag\_()

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

## Функция для сохранения информации о столкновениях

## Аргументы

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

## 7.10.3.8 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

## 7.10.3.9 IsIntersecting\_()

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

Функция для проверки наличия столкновения с other.

## Аргументы

other	указатель на второй объект
-------	----------------------------

## Возвращает

true, если объект пересекается с other, иначе false

## 7.10.3.10 IsNear\_()

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

Функция для проверки, находится ли other рядом с данным объектом

## Аргументы

other	указатель на второй объект
-------	----------------------------

## Возвращает

true, если other рядом, иначе false

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

##### 7.10.4.1 CheckCollisions

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

Функция для вычисления столкновений в мире

Аргументы

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

Объявления и описания членов классов находятся в файлах:

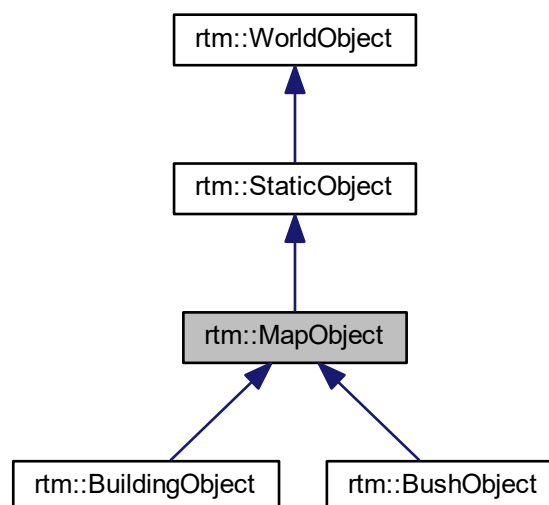
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DynamicObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/DynamicObject.cpp

#### 7.11 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)  
Конструктор с использованием уже готового спрайта
- [MapObject](#) (std::string const &filename, int column, int row, float angle)  
Конструктор из файла
- virtual [~MapObject](#) ()=default  
Деструктор по умолчанию

## Дополнительные унаследованные члены

## 7.11.1 Подробное описание

Класс статического объекта карты

## 7.11.2 Конструктор(ы)

## 7.11.2.1 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

## 7.11.2.2 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:

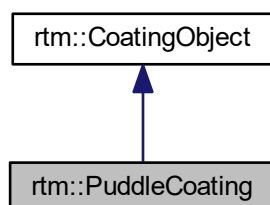
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/MapObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/MapObject.cpp

## 7.12 Класс rtm::PuddleCoating

Класс, описывающий лужи

```
#include <PuddleCoating.h>
```

Граф наследования: rtm::PuddleCoating:



## Открытые члены

- [PuddleCoating](#) ()  
Конструктор по умолчанию
- [PuddleCoating](#) (cocos2d::Sprite \*const sprite, int column, int row, [AngleType](#) angle)  
Конструктор с использованием уже готового спрайта
- [PuddleCoating](#) (std::string const &filename, int column, int row, [AngleType](#) angle)  
Конструктор из файла
- [PuddleCoating](#) (size\_t id, int column, int row, [AngleType](#) angle)  
Конструктор стандартного лужи
- virtual [~PuddleCoating](#) ()=default  
Деструктор по умолчанию

## Дополнительные унаследованные члены

### 7.12.1 Подробное описание

Класс, описывающий лужи

### 7.12.2 Конструктор(ы)

#### 7.12.2.1 PuddleCoating() [1/3]

```
rtm::PuddleCoating::PuddleCoating (
    cocos2d::Sprite *const sprite,
    int column,
    int row,
    AngleType angle )
```

Конструктор с использованием уже готового спрайта

Аргументы

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

#### 7.12.2.2 PuddleCoating() [2/3]

```
rtm::PuddleCoating::PuddleCoating (
    std::string const & filename,
    int column,
    int row,
    AngleType angle )
```

Конструктор из файла

Аргументы

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

## 7.12.2.3 PuddleCoating() [3/3]

```
rtm::PuddleCoating::PuddleCoating (
    size_t id,
    int column,
    int row,
    AngleType angle )
```

Конструктор стандартного лужи

Аргументы

id	номер стандартной лужи
column	колонка, в которой необходимо отрисовать лужу
row	строка, в которой необходимо отрисовать лужу
angle	угол поворота лужи

Объявления и описания членов классов находятся в файлах:

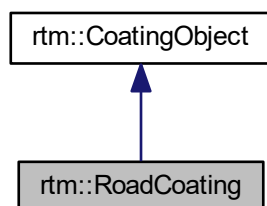
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/PuddleCoating.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/PuddleCoating.cpp

## 7.13 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)

Конструктор с использованием уже готового спрайта

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

Конструктор из файла

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

Конструктор стандартного дороги

- `virtual ~RoadCoating ()=default`

Деструктор по умолчанию

## Закрытые статические члены

- `static float GetClassResistance_ (CoatingType type)`

Функция для получения коэффициента сопротивления на стандартной дороге по номеру

- `static Directions const & GetClassDirections_ (size_t id)`

Функция для получения доступных направлений стандартной дороги по номеру

## Дополнительные унаследованные члены

### 7.13.1 Подробное описание

Класс, описывающий дороги

### 7.13.2 Конструктор(ы)

#### 7.13.2.1 `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

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

## 7.13.2.2 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

## 7.13.2.3 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
column	колонка, в которой необходимо отрисовать дорогу
row	строка, в которой необходимо отрисовать дорогу
angle	угол поворота дороги

## 7.13.3 Методы

### 7.13.3.1 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

### 7.13.3.2 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

## 7.14 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

### 7.14.1 Подробное описание

Структура, описывающая параметры точки генерации объектов

Объявления и описания членов структуры находятся в файле:

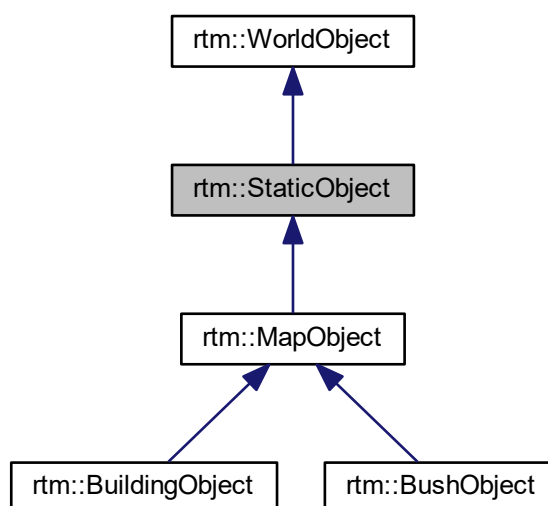
- `C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/General.h`

## 7.15 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)`  
Конструктор с использованием уже готового спрайта
- `StaticObject(std::string const &filename, float x, float y, float angle)`  
Конструктор из файла
- `virtual ~StaticObject()=default`  
Деструктор по умолчанию

## Дополнительные унаследованные члены

### 7.15.1 Подробное описание

Класс статического объекта (который не обновляется)

### 7.15.2 Конструктор(ы)

#### 7.15.2.1 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

#### 7.15.2.2 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:



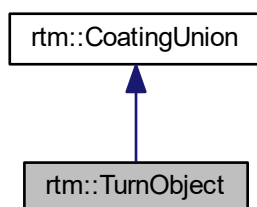
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/StaticObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/StaticObject.cpp

## 7.16 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)  
Конструктор по размерам
- virtual [~TurnObject](#) ()=default  
Деструктор по умолчанию
- bool [IsRight](#) () const  
Функция для получения типа поворота
- [AngleType GetAngle](#) () const  
Функция для получения угла поворота объекта

### Открытые статические члены

- static [CoatingMatrix RightTurnMatrix](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)  
Функция для получения матрицы правого поворота
- static [CoatingMatrix LeftTurnMatrix](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)  
Функция для получения матрицы левого поворота

## Закрытые данные

- `bool isRight_`  
Тип поворота
- `AngleType angle_`  
Угол поворота объекта

## Дополнительные унаследованные члены

### 7.16.1 Подробное описание

#### Класс поворота дороги

### 7.16.2 Конструктор(ы)

#### 7.16.2.1 TurnObject()

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

#### Конструктор по размерам

##### Аргументы

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

### 7.16.3 Методы

#### 7.16.3.1 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

#### 7.16.3.2 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

#### 7.16.3.3 IsRight()

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

Функция для получения типа поворота

Возвращает

true, если правый поворот, false, если левый

#### 7.16.3.4 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:

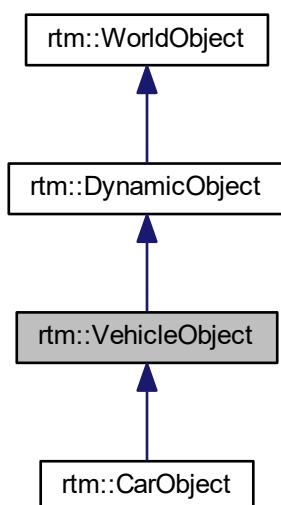
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/TurnObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/TurnObject.cpp

### 7.17 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`)  
Конструктор с использованием уже готового спрайта
- `VehicleObject` (`std::string const &filename`, `int column`, `int row`, `float angle`, `float maxSpeed`, `float acceleration`, `float deceleration`)  
Конструктор из файла
- `virtual ~VehicleObject` ()=default  
Деструктор по умолчанию
- `virtual void Update` (`WorldController *const world`) override  
Функция обновления

## Защищенные члены

- `bool MoveForward_` ()  
Функция для сообщения о необходимости начать движение
- `bool Stop_` ()  
Функция для сообщения о необходимости остановиться
- `bool Rotate_` (`float angle=ANGLE_RIGHT`)  
Функция для сообщения о необходимости повернуть
- `bool ChangeLine_` (`bool isRight=LEFT`)  
Функция для сообщения о необходимости перестроиться
- `bool IsMovement_` () const  
Функция, сообщающая о движении объекта
- `bool IsRotation_` () const  
Функция, сообщающая о повороте объекта
- `bool IsLineChanging_` () const  
Функция, сообщающая о перестроении объекта
- `bool IsBraking_` () const  
Функция, сообщающая о торможении объекта перед светофором и т.д.
- `float GetMaxSpeed_` () const  
Функция для получения максимальной скорости
- `float GetFinalSpeed_` () const  
Функция для получения финальной скорости (к которой объект будет стремиться)
- `void SetFinalSpeed_` (`float speed`)  
Функция для установки финальной скорости (к которой объект будет стремиться)
- `void SetBrakingFactor_` (`float factor`)  
Функция для установки коэффициента торможения
- `void StopAtDistance_` (`float distance`)  
Функция для установки тормозного пути (объект будет пытаться остановиться за данную дистанцию)

## Зрение у машин

Функции для просмотра окружающего мира

- `CoatingObject * CheckForwardCoating_` (`WorldController *const world`, `int delta=1`)  
Функция для получения следующего по ходу движения покрытия
- `CoatingUnion * CheckForwardCoatingUnion_` (`WorldController *const world`, `int delta=1`)

- Функция для получения следующего по ходу движения объединения покрытий
- `DynamicObject * CheckForwardArea_ (WorldController *const world, float radius, float angle, float angleShift)`
- Функция для проверки области видимости спереди
- `DynamicObject * CheckMovingArea_ (WorldController *const world)`
- Функция для проверки области видимости во время движения по прямой
- `DynamicObject * CheckTurnArea_ (WorldController *const world, bool isRight)`
- Функция для проверки области видимости перед поворотом
- `DynamicObject * CheckRotationArea_ (WorldController *const world)`
- Функция для проверки области видимости во время поворота
- `DynamicObject * CheckCrossroadArea_ (WorldController *const world)`
- Функция для проверки области видимости перед нерегулируемым перекрестком
- `DynamicObject * CheckLineChangingArea_ (WorldController *const world)`
- Функция для проверки области видимости перед перестроением

## Маневры

Функции выполняющиеся во время маневров

- `virtual void BeforeMoving_ (WorldController *const world)`  
Функция, выполняющаяся непосредственно перед перемещением объекта
- `virtual void AfterMoving_ (WorldController *const world)`  
Функция, выполняющаяся непосредственно после перемещением объекта
- `virtual bool MovementStart_ (WorldController *const world)`  
Функция, выполняющаяся перед началом движения
- `virtual bool MovementTick_ (WorldController *const world)`  
Функция, выполняющаяся во время движения
- `virtual bool MovementEnd_ (WorldController *const world)`  
Функция, выполняющаяся после движения (перед началом остановки)
- `virtual bool RotationStart_ (WorldController *const world)`  
Функция, выполняющаяся перед поворотом
- `virtual bool RotationTick_ (WorldController *const world)`  
Функция, выполняющаяся во время поворота
- `virtual bool RotationEnd_ (WorldController *const world)`  
Функция, выполняющаяся после поворота
- `virtual bool LineChangingStart (WorldController *const world)`  
Функция, выполняющаяся перед перестроением
- `virtual bool LineChangingTick_ (WorldController *const world)`  
Функция, выполняющаяся во время перестроения
- `virtual bool LineChangingEnd_ (WorldController *const world)`  
Функция, выполняющаяся после перестроения

## Закрытые члены

- `void LineChanging_ (WorldController *const world)`  
Функция, выполняющая различные этапы при перестроении
- `void Rotation_ (WorldController *const world)`  
Функция, выполняющая различные этапы при повороте
- `void Movement_ (WorldController *const world)`  
Функция, выполняющая различные этапы при движении
- `void SpeedChanging_ (WorldController *const world)`  
Функция, выполняющая изменение скорости в зависимости от ускорения, тормозного пути и т.д.
- `void SmoothBrakingCounter (WorldController *const world)`  
Функция, выполняющая декрементирование тормозного пути (если задан)

## 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)

## 7.17.1 Подробное описание

Класс транспорта (динамического объекта карты)

## 7.17.2 Конструктор(ы)

7.17.2.1 `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

## 7.17.2.2 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

## 7.17.3 Методы

## 7.17.3.1 Update()

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

## Функция обновления



Аргументы

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

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

#### 7.17.3.2 `MoveForward_()`

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

Функция для сообщения о необходимости начать движение

Возвращает

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

#### 7.17.3.3 `Stop_()`

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

Функция для сообщения о необходимости остановиться

Возвращает

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

#### 7.17.3.4 `Rotate_()`

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

Функция для сообщения о необходимости повернуть

Возвращает

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

#### 7.17.3.5 ChangeLine\_()

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

Функция для сообщения о необходимости перестроиться

Возвращает

true, если возможно перестроиться, иначе false

#### 7.17.3.6 IsMovement\_()

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

Функция, сообщающая о движении объекта

Возвращает

true, если объект движется, иначе false

#### 7.17.3.7 IsRotation\_()

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

Функция, сообщающая о повороте объекта

Возвращает

true, если объект поворачивает, иначе false

#### 7.17.3.8 IsLineChanging\_()

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

Функция, сообщающая о перестроении объекта

Возвращает

true, если объект перестраивается, иначе false

## 7.17.3.9 IsBraking\_()

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

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

Возвращает

true, если объект тормозит, иначе false

## 7.17.3.10 GetMaxSpeed\_()

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

Функция для получения максимальной скорости

Возвращает

максимальная скорость

## 7.17.3.11 GetFinalSpeed\_()

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

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

Возвращает

конечная скорость

## 7.17.3.12 SetFinalSpeed\_()

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

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

Аргументы

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

## 7.17.3.13 SetBrakingFactor\_()

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

Функция для установки коэффициента торможения

Аргументы

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

## 7.17.3.14 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
----------	---

## 7.17.3.15 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

## 7.17.3.16 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

## 7.17.3.17 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

## 7.17.3.18 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

## 7.17.3.19 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

## 7.17.3.20 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

## 7.17.3.21 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

## 7.17.3.22 CheckLineChangingArea\_()

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

Функция для проверки области видимости перед перестроением

Аргументы

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

Возвращает

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

## 7.17.3.23 BeforeMoving\_()

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

Функция, выполняющаяся непосредственно перед перемещением объекта

Аргументы

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

## 7.17.3.24 AfterMoving\_()

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

Функция, выполняющаяся непосредственно после перемещением объекта

Аргументы

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

## 7.17.3.25 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](#).

## 7.17.3.26 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](#).

#### 7.17.3.27 `MovementEnd_()`

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

Функция, выполняющаяся после движения (перед началом остановки)

Аргументы

world	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](#).

#### 7.17.3.28 `RotationStart_()`

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

Функция, выполняющаяся перед поворотом

Аргументы

world	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)

#### 7.17.3.29 `RotationTick_()`

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

Функция, выполняющаяся во время поворота

## Аргументы

world	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)

7.17.3.30 `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)

7.17.3.31 `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`.

7.17.3.32 `LineChangingTick_()`

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

Функция, выполняющаяся во время перестроения

Аргументы

<code>world</code>	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)

7.17.3.33 `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)

7.17.3.34 `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
--------------------	--

## 7.17.3.35 Rotation\_()

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

Функция, выполняющая различные этапы при повороте

Аргументы

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

## 7.17.3.36 Movement\_()

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

Функция, выполняющая различные этапы при движении

Аргументы

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

## 7.17.3.37 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
-------	--

## 7.17.3.38 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

## 7.18 Класс rtm::WorldController

Класс контроллера карты, связующее звено всех объектов

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

## Открытые члены

- [WorldController](#) ()  
Конструктор по умолчанию
- [WorldController](#) ([WorldScene](#) \*const scene)  
Конструктор без загрузки какой-либо карты
- [WorldController](#) ([WorldScene](#) \*const scene, std::string const &filename)  
Конструктор с загрузкой карты
- [WorldController](#) ([WorldScene](#) \*const scene, size\_t mapNumber)  
Конструктор с загрузкой карты
- void [Update](#) (float time)  
Функция обновления
- cocos2d::Layer \* [GetLayer](#) () const  
Функция для получения основного слоя сцены (на нём вся движуха)
- size\_t [GetColumnsCount](#) () const  
Функция для получения количества столбцов карты
- size\_t [GetRowCount](#) () const  
Функция для получения количества строк карты
- float [GetDeltaTime](#) () const  
Функция для получения последней разницей между обновлениями
- float [GetTimeFactor](#) () const  
Функция для получения коэффициента ускорения времени
- [CoatingObject](#) \* [GetCoatingObject](#) (int column, int row)  
Функция для получения объекта в определенной клетке
- [CoatingUnion](#) \* [GetCoatingUnion](#) (int column, int row)  
Функция для получения объединения покрытий в определенной клетке
- [StaticObject](#) \* [GetStaticObject](#) (int column, int row)  
Функция для получения статического объекта в определенной клетке
- std::vector< [DynamicShared](#) > & [GetDynamicObjects](#) ()  
Функция для получения массива движущихся объектов
- bool [IsPause](#) ()  
Функция сообщает, происходят ли обновления
- bool [IsCorrectColumn](#) (int column)

- Функция проверяет корректность столбца
- bool [IsCorrectRow](#) (int row)
- Функция проверяет корректность строки
- bool [IsAllowableColumn](#) (int column)
- Функция проверяет, можно ли двигаться в столбце
- bool [IsAllowableRow](#) (int row)
- Функция проверяет, можно ли двигаться в строке
- bool [IsVisibleColumn](#) (int column)
- Функция проверяет видимость столбца
- bool [IsVisibleRow](#) (int row)
- Функция проверяет видимость строки
- void [SetTimeFactor](#) (float factor)
- Функция для установки коэффициента ускорения времени
- bool [LoadMap](#) (std::string const &filename)
- Функция для загрузки карты из файла
- bool [LoadMap](#) (size\_t number)
- Функция для загрузки карты по номеру
- 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)
- Функция проверяет доступность зоны для генерации статических объектов и объектов объекта
- bool [GenerateObject\\_](#) (uint8\_t \*params, uint8\_t count)
- Функция для парсинга параметров и генерации объектов
- bool [AddCoatingUnion\\_](#) (int column, int row, [CoatingUnionShared](#) coatingUnion)
- Функция для генерации объединения покрытий
- bool [AddDriveway\\_](#) ([CoatingType](#) type, int column, int row, size\_t width, size\_t height, [AngleType](#) angle)
- Функция для генерации прямой односторонней дороги
- bool [AddCrossroad\\_](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, size\_t controlUnitType=0)
- Функция для генерации перекрестка
- bool [AddTCrossroad\\_](#) ([CoatingType](#) type, int column, int row, [LinesCounts](#) linesCounts, [AngleType](#) nullDirection, size\_t controlUnitType=0)
- Функция для генерации т-образного перекрестка
- bool [AddLeftTurt\\_](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)
- Функция для генерации левого поворота
- bool [AddRightTurt\\_](#) ([CoatingType](#) type, int column, int row, size\_t linesCount, [AngleType](#) angle)

- `bool AddControlUnit_ (ControlUnitShared controlUnit)`  
Функция для генерации правого поворота
- `bool AddStaticObject_ (int column, int row, StaticShared staticObject)`  
Функция для добавления управляющего блока в общий массив (для обновлений)
- `bool AddBuilding_ (size_t type, int column, int row, float angle)`  
Функция для генерации статического объекта
- `bool AddBush_ (size_t type, int column, int row, float angle)`  
Функция для генерации строения
- `bool AddDynamicObject_ (int column, int row, DynamicShared dynamicObject)`  
Функция для генерации динамического объекта
- `bool AddCar_ (size_t type, int column, int row, float angle)`  
Функция для генерации машины
- `size_t GetVectorColumn_ (int column)`  
Функция для получения столбца в массиве
- `size_t GetVectorRow_ (int row)`  
Функция для получения строки в массиве
- `int GetRealColumn_ (size_t column)`  
Функция для получения столбца объекта
- `int GetRealRow_ (size_t row)`  
Функция для получения строки объекта
- `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 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 автоматического 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

### 7.18.1 Подробное описание

Класс контроллера карты, связующее звено всех объектов

### 7.18.2 Конструктор(ы)

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

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

Конструктор без загрузки какой-либо карты

Аргументы

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

#### 7.18.2.2 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

## 7.18.2.3 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

## 7.18.3 Методы

## 7.18.3.1 Update()

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

## Функция обновления

## Аргументы

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

## 7.18.3.2 GetLayer()

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

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

## Возвращает

основной слой

### 7.18.3.3 GetColumnsCount()

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

Функция для получения количества столбцов карты

Возвращает

количество столбцов

### 7.18.3.4 GetRowCount()

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

Функция для получения количества строк карты

Возвращает

количество строк

### 7.18.3.5 GetDeltaTime()

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

Функция для получения последней разницей между обновлениями

Возвращает

разница во времени между обновлениями (в секундах)

### 7.18.3.6 GetTimeFactor()

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

Функция для получения коэффициента ускорения времени

Возвращает

коэффициент ускорения времени (1 - реальная скорость)

## 7.18.3.7 GetCoatingObject()

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

Функция для получения объекта в определенной клетке

Возвращает

указатель на объект объекта

## 7.18.3.8 GetCoatingUnion()

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

Функция для получения объединения покрытий в определенной клетке

Возвращает

указатель на объект объединения покрытий

## 7.18.3.9 GetStaticObject()

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

Функция для получения статического объекта в определенной клетке

Возвращает

указатель на статический объект

## 7.18.3.10 GetDynamicObjects()

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

Функция для получения массива движущихся объектов

Возвращает

массив движущихся объектов

## 7.18.3.11 IsPause()

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

Функция сообщает, происходят ли обновления

Возвращает

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

## 7.18.3.12 IsCorrectColumn()

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

Функция проверяет корректность столбца

Аргументы

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

Возвращает

true, если столбец корректный, иначе false

## 7.18.3.13 IsCorrectRow()

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

Функция проверяет корректность строки

Аргументы

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

Возвращает

true, если строка корректная, иначе false

7.18.3.14 `IsAllowableColumn()`

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

Функция проверяет, можно ли двигаться в столбце

Аргументы

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

Возвращает

`true`, если в столбце можно двигаться, иначе `false`

7.18.3.15 `IsAllowableRow()`

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

Функция проверяет, можно ли двигаться в строке

Аргументы

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

Возвращает

`true`, если в строке можно двигаться, иначе `false`

7.18.3.16 `IsVisibleColumn()`

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

Функция проверяет видимость столбца

Аргументы

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

Возвращает

true, если столбец виден, иначе false

#### 7.18.3.17 IsVisibleRow()

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

Функция проверяет видимость строки

Аргументы

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

Возвращает

true, если строка видна, иначе false

#### 7.18.3.18 SetTimeFactor()

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

Функция для установки коэффициента ускорения времени

Аргументы

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

#### 7.18.3.19 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
----------	------------------------------

## 7.18.3.20 LoadMap() [2/2]

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

Функция для загрузки карты по номеру

Аргументы

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

## 7.18.3.21 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

## 7.18.3.22 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

## 7.18.3.23 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

## 7.18.3.24 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

### 7.18.3.25 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

### 7.18.3.26 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

#### 7.18.3.27 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

#### 7.18.3.28 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

#### 7.18.3.29 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

#### 7.18.3.30 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

#### 7.18.3.31 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

#### 7.18.3.32 AddBush\_()

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

Функция для генерации куста

Аргументы

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

Возвращает

true, если объект получилось сгенерировать, иначе false

#### 7.18.3.33 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

## 7.18.3.34 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

## 7.18.3.35 GetVectorColumn\_()

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

## Функция для получения столбца в массиве

## Аргументы

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

Возвращает

столбец в массиве

#### 7.18.3.36 GetVectorRow\_()

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

Функция для получения строки в массиве

Аргументы

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

Возвращает

строка в массиве

#### 7.18.3.37 GetRealColumn\_()

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

Функция для получения столбца объекта

Аргументы

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

Возвращает

столбец объекта

#### 7.18.3.38 GetRealRow\_()

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

Функция для получения строки объекта

## Аргументы

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

## Возвращает

строка объекта

Объявления и описания членов классов находятся в файлах:

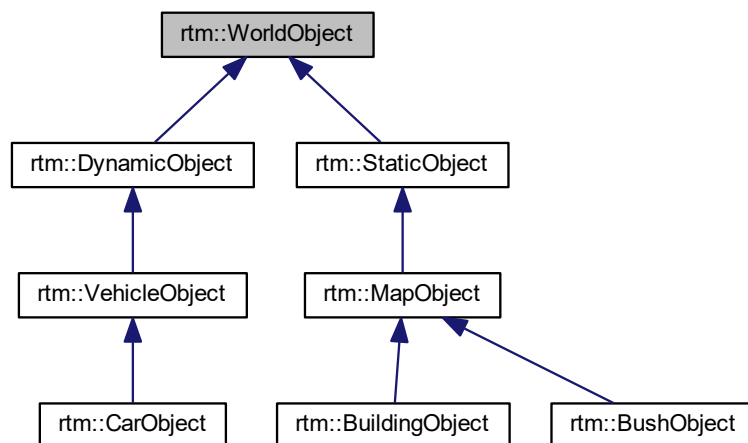
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldController.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldController.cpp

## 7.19 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)  
Конструктор с использованием уже готового спрайта
- [WorldObject](#) (std::string const &filename, float x, float y, float angle)  
Конструктор из файла

- virtual `~WorldObject ()`=default  
Деструктор по умолчанию
- cocos2d::Sprite \* `GetSprite ()` const  
Функция для получения спрайта
- float `GetX_ ()` const  
Функция для получения абсциссы
- float `GetY_ ()` const  
Функция для получения ординаты
- float `GetAngle ()` const  
Функция для получения угла поворота
- float `GetWidth ()` const  
Функция для получения ширины
- float `GetHeight ()` const  
Функция для получения высоты

### Защищенные члены

- void `SetSprite_ (cocos2d::Sprite *const sprite)`  
Функция для установки спрайта
- void `SetX_ (float x)`  
Функция для установки абсциссы
- void `SetY_ (float y)`  
Функция для установки ординаты
- void `SetAngle_ (float angle)`  
Функция для установки угла поворота
- void `SetWidth_ (float width)`  
Функция для установки ширины
- void `SetHeight_ (float height)`  
Функция для установки высоты
- 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)  
Функция для установки абсциссы спрайта
- void `SetSpriteY_` (float y)  
Функция для установки ординаты спрайта
- void `SetSpriteAngle_` (float angle)  
Функция для установки угла поворота спрайта
- void `SetSpriteWidth_` (float width)  
Функция для установки ширины спрайта
- void `SetSpriteHeight_` (float height)  
Функция для установки высоты спрайта

## 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

## 7.19.1 Подробное описание

Класс объекта мира (родитель всех условно объемных объектов)

## 7.19.2 Конструктор(ы)

## 7.19.2.1 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

## 7.19.2.2 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

## 7.19.3 Методы

## 7.19.3.1 GetSprite()

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

Функция для получения спрайта

Возвращает

указатель на спрайт

## 7.19.3.2 GetX\_()

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

Функция для получения абсциссы

Возвращает

абсцисса

### 7.19.3.3 `GetY_()`

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

Функция для получения ординаты

Возвращает

ордината

### 7.19.3.4 `GetAngle()`

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

Функция для получения угла поворота

Возвращает

угол поворота

### 7.19.3.5 `GetWidth()`

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

Функция для получения ширины

Возвращает

ширина

### 7.19.3.6 `GetHeight()`

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

Функция для получения высоты

Возвращает

высота

### 7.19.3.7 `SetSprite_()`

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

Функция для установки спрайта

Аргументы

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

#### 7.19.3.8 SetX\_()

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

Функция для установки абсциссы

Аргументы

x	абсцисса
---	----------

#### 7.19.3.9 SetY\_()

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

Функция для установки ординаты

Аргументы

y	ордината
---	----------

#### 7.19.3.10 SetAngle\_()

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

Функция для установки угла поворота

Аргументы

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

## 7.19.3.11 SetWidth\_()

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

Функция для установки ширины

Аргументы

width	ширина
-------	--------

## 7.19.3.12 SetHeight\_()

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

Функция для установки высоты

Аргументы

height	высота
--------	--------

## 7.19.3.13 SetSpriteX\_()

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

Функция для установки абсциссы спрайта

Аргументы

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

## 7.19.3.14 SetSpriteY\_()

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

Функция для установки ординаты спрайта

Аргументы

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

#### 7.19.3.15 SetSpriteAngle\_()

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

Функция для установки угла поворота спрайта

Аргументы

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

#### 7.19.3.16 SetSpriteWidth\_()

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

Функция для установки ширины спрайта

Аргументы

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

#### 7.19.3.17 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:

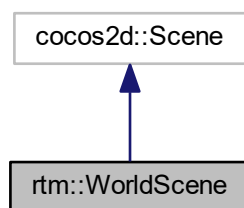
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldObject.h
- C:/Users/Vladimir/Documents/Visual Studio 2017/Projects/RTM/Classes/WorldObject.cpp

## 7.20 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  
Функция для инициализации полей
- virtual void `update(float time)` override  
Функция для обновления сцены
- cocos2d::Layer \* `GetMainLayer()` const  
Функция для получения основного слоя, на котором находятся объекты

Функции для установки фона

- void `SetBackground(std::string const &filename)`  
Функции для установки фона из файла
- void `SetBackground(size_t number)`  
Функции для установки фона по номеру

Открытые статические члены

- static `WorldScene * Create()`  
Конструктор класса, поддерживающий RAII.

## 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\\_](#) ()  
Функция для получения контроллера данной сцены

## Закрытые статические члены

- 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.

#### 7.20.1 Подробное описание

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

#### 7.20.2 Методы

##### 7.20.2.1 Create()

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

Конструктор класса, поддерживающий RAII.

Возвращает

указатель на созданный объект

## 7.20.2.2 init()

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

Функция для инициализации полей

Возвращает

true в случае успешной инициализации, иначе false

## 7.20.2.3 update()

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

Функция для обновления сцены

Аргументы

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

## 7.20.2.4 GetMainLayer()

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

Функция для получения основного слоя, на котором находятся объекты

Возвращает

основной слой

## 7.20.2.5 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)
----------	--

## 7.20.2.6 SetBackground() [2/2]

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

Функции для установки фона по номеру

Аргументы

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

## 7.20.2.7 GetMap\_()

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

Функция для получения контроллера данной сцены

Возвращает

контроллер сцены

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, [125](#)
- AddBush\_
  - rtm::WorldController, [126](#)
- AddCar\_
  - rtm::WorldController, [127](#)
- AddCoatingUnion\_
  - rtm::WorldController, [122](#)
- AddControlUnit\_
  - rtm::WorldController, [125](#)
- AddCrossroad\_
  - rtm::WorldController, [123](#)
- AddDriveway\_
  - rtm::WorldController, [122](#)
- AddDynamicObject\_
  - rtm::WorldController, [126](#)
- AddLeftTurt\_
  - rtm::WorldController, [124](#)
- AddRightTurt\_
  - rtm::WorldController, [124](#)
- AddStaticObject\_
  - rtm::WorldController, [125](#)
- AddTCrossroad\_
  - rtm::WorldController, [123](#)
- AfterMoving\_
  - rtm::VehicleObject, [105](#)
- Allowed
  - rtm, [21](#)
- AngleToAngleType
  - rtm, [27](#)
- AngleToDirection
  - rtm, [28](#)
- AngleType
  - rtm, [19](#)
- AngleTypeToAngle
  - rtm, [28](#)
- AngleTypeToDirection
  - rtm, [28](#)
- AppDelegate, [35](#)
  - applicationDidFinishLaunching, [36](#)
- applicationDidFinishLaunching
  - AppDelegate, [36](#)
- BeforeMoving\_
  - rtm::VehicleObject, [105](#)
- BuildingObject
  - rtm::BuildingObject, [37](#), [38](#)
- BushObject
  - rtm::BushObject, [39](#), [40](#)
- CARS\_ACCELERATIONS
  - rtm, [33](#)
- CARS\_MAX\_SPEEDS
  - rtm, [32](#)
- COATING\_INDEXES
  - rtm, [31](#)
- CarObject
  - rtm::CarObject, [43](#), [44](#)
- CellToPixel
  - rtm, [27](#)
- CenterIsCrossed
  - rtm, [25](#)
- ChangeLine\_
  - rtm::VehicleObject, [99](#)
- CheckCoatingAhead\_
  - rtm::CarObject, [47](#)
- CheckCoatingUnionAhead\_
  - rtm::CarObject, [47](#)
- CheckCollisions
  - rtm, [23](#)
    - rtm::DynamicObject, [80](#)
- CheckCrossroadArea\_
  - rtm::VehicleObject, [104](#)
- CheckForwardArea\_
  - rtm::VehicleObject, [103](#)
- CheckForwardCoating\_
  - rtm::VehicleObject, [102](#)
- CheckForwardCoatingUnion\_
  - rtm::VehicleObject, [102](#)
- CheckLineChangingArea\_
  - rtm::VehicleObject, [105](#)
- CheckMovingArea\_
  - rtm::VehicleObject, [103](#)
- CheckRoadAhead\_
  - rtm::CarObject, [47](#)
- CheckRotationArea\_
  - rtm::VehicleObject, [104](#)
- CheckTurnArea\_
  - rtm::VehicleObject, [104](#)
- Closed
  - rtm, [21](#)
- CoatingObject
  - rtm::CoatingObject, [50](#)
- CoatingType
  - rtm, [21](#)
- CoatingUnion
  - rtm::CoatingUnion, [55](#)
- CoatingUnionType
  - rtm, [20](#)

- ControlUnit
  - rtm::ControlUnit, [60](#)
- CountDeceleration
  - rtm, [30](#)
- CountLength\_
  - rtm::DrivewayObject, [72](#)
- CountLines\_
  - rtm::DrivewayObject, [73](#)
- Create
  - rtm::WorldScene, [139](#)
- CrossroadMatrix
  - rtm::CrossroadObject, [65](#)
- CrossroadObject
  - rtm::CrossroadObject, [64](#), [65](#)
- CrossroadSignals
  - rtm, [18](#)
- DEFAULT\_CROSSROAD\_SIGNALS
  - rtm, [31](#)
- DEFAULT DIRECTIONS\_SIGNAL\_SPRITES
  - rtm, [31](#)
- DirectionSignalIndex
  - rtm, [20](#)
- DirectionSignals
  - rtm, [18](#)
- DirectionToAngle
  - rtm, [29](#)
- DirectionToAngleType
  - rtm, [29](#)
- DirectionType
  - rtm, [20](#)
- Directions
  - rtm, [18](#)
- DirectionsSignalSprites
  - rtm, [19](#)
- DistanceToNextCenter
  - rtm, [24](#)
- Down
  - rtm, [20](#)
- Downward
  - rtm, [20](#)
- DrivewayMatrix
  - rtm::DrivewayObject, [70](#)
- DrivewayObject
  - rtm::DrivewayObject, [69](#)
- DynamicObject
  - rtm::DynamicObject, [75](#), [76](#)
- Forbidden
  - rtm, [21](#)
- GenerateObject\_
  - rtm::WorldController, [121](#)
- GetAngle
  - rtm::TurnObject, [94](#)
  - rtm::WorldObject, [133](#)
- GetClassAcceleration\_
  - rtm::CarObject, [48](#)
- GetClassDirections\_
  - rtm::RoadCoating, [88](#)
- GetClassMaxSpeed\_
  - rtm::CarObject, [47](#)
- GetClassResistance\_
  - rtm::RoadCoating, [87](#)
- GetCoatingObject
  - rtm::CoatingUnion, [56](#)
  - rtm::WorldController, [116](#)
- GetCoatingUnion
  - rtm::WorldController, [117](#)
- GetColumn\_
  - rtm::CoatingUnion, [58](#)
- GetColumnsCount
  - rtm::WorldController, [115](#)
- GetControlUnit
  - rtm::CrossroadObject, [67](#)
- GetDeltaTime
  - rtm::WorldController, [116](#)
- GetDynamicObjects
  - rtm::WorldController, [117](#)
- GetFilename
  - rtm, [29](#)
- GetFinalSpeed\_
  - rtm::VehicleObject, [101](#)
- GetHeight
  - rtm::CoatingUnion, [56](#)
  - rtm::WorldObject, [133](#)
- GetLastDelta
  - rtm::DynamicObject, [76](#)
- GetLayer
  - rtm::WorldController, [115](#)
- GetLength
  - rtm::CoatingUnion, [57](#)
  - rtm::DrivewayObject, [70](#)
- GetLinesCount
  - rtm::DrivewayObject, [70](#)
- GetMainLayer
  - rtm::WorldScene, [140](#)
- GetMap\_
  - rtm::WorldScene, [141](#)
- GetMaxSpeed\_
  - rtm::VehicleObject, [101](#)
- GetNullDirection
  - rtm::CrossroadObject, [66](#)
- GetRealColumn\_
  - rtm::WorldController, [128](#)
- GetRealRow\_
  - rtm::WorldController, [128](#)
- GetResistance
  - rtm::CoatingObject, [51](#)
- GetRow\_
  - rtm::CoatingUnion, [58](#)
- GetRowsCount
  - rtm::WorldController, [116](#)
- GetSignal
  - rtm::ControlUnit, [61](#)
- GetSpeed
  - rtm::DynamicObject, [76](#)

- GetSprite
  - rtm::CoatingObject, [51](#)
  - rtm::WorldObject, [132](#)
- GetStaticObject
  - rtm::WorldController, [117](#)
- GetTimeFactor
  - rtm::WorldController, [116](#)
- GetType
  - rtm::CoatingUnion, [55](#)
- GetVectorColumn\_
  - rtm::WorldController, [127](#)
- GetVectorRow\_
  - rtm::WorldController, [128](#)
- GetWidth
  - rtm::CoatingUnion, [56](#)
  - rtm::WorldObject, [133](#)
- GetX\_
  - rtm::WorldObject, [132](#)
- GetY\_
  - rtm::WorldObject, [132](#)
- HasCollision
  - rtm::DynamicObject, [77](#)
- HasDirection
  - rtm::CoatingObject, [51](#)
- InCenter
  - rtm, [24](#)
- init
  - rtm::WorldScene, [139](#)
- IsAllowableColumn
  - rtm::WorldController, [118](#)
- IsAllowableRow
  - rtm::WorldController, [119](#)
- IsBeholding\_
  - rtm::DynamicObject, [78](#)
- IsBraking\_
  - rtm::VehicleObject, [100](#)
- IsCorrectColumn
  - rtm::CoatingUnion, [57](#)
  - rtm::WorldController, [118](#)
- IsCorrectRow
  - rtm::CoatingUnion, [57](#)
  - rtm::WorldController, [118](#)
- IsDirectionAvailable
  - rtm::CoatingObject, [52](#)
- IsEmpty\_
  - rtm::WorldController, [121](#)
- IsIntersecting\_
  - rtm::DynamicObject, [79](#)
- isLeftLine
  - rtm::DrivewayObject, [71](#), [72](#)
- IsLineChanging\_
  - rtm::VehicleObject, [100](#)
- IsMovement\_
  - rtm::VehicleObject, [100](#)
- IsNear\_
  - rtm::DynamicObject, [79](#)
- IsNearOthers
  - rtm::DynamicObject, [77](#)
- IsPause
  - rtm::WorldController, [117](#)
- IsRight
  - rtm::TurnObject, [93](#)
- isRightLine
  - rtm::DrivewayObject, [71](#)
- IsRotation\_
  - rtm::VehicleObject, [100](#)
- IsVisibleColumn
  - rtm::WorldController, [119](#)
- IsVisibleRow
  - rtm::WorldController, [120](#)
- Left
  - rtm, [20](#)
- LeftTurnMatrix
  - rtm::TurnObject, [93](#)
- Leftward
  - rtm, [20](#)
- LineChanging\_
  - rtm::VehicleObject, [109](#)
- LineChangingEnd\_
  - rtm::VehicleObject, [109](#)
- LineChangingStart
  - rtm::CarObject, [45](#)
  - rtm::VehicleObject, [108](#)
- LineChangingTick\_
  - rtm::VehicleObject, [108](#)
- LinesCounts
  - rtm, [18](#)
- LoadMap
  - rtm::WorldController, [120](#), [121](#)
- MapObject
  - rtm::MapObject, [81](#)
- MoveForward\_
  - rtm::VehicleObject, [99](#)
- Movement\_
  - rtm::VehicleObject, [110](#)
- MovementEnd\_
  - rtm::CarObject, [45](#)
  - rtm::VehicleObject, [107](#)
- MovementStart\_
  - rtm::CarObject, [44](#)
  - rtm::VehicleObject, [106](#)
- MovementTick\_
  - rtm::CarObject, [45](#)
  - rtm::VehicleObject, [106](#)
- NEAR\_DELTA
  - rtm, [31](#)
- NormalizeAngle
  - rtm, [26](#)
- operator bool
  - rtm::ControlUnit, [61](#)
- PixelToCell

- rtm, 26
- PuddleCoating
  - rtm::PuddleCoating, 84
- ROADS\_DIRECTIONS
  - rtm, 32
- ROADS\_RESISTANCES
  - rtm, 32
- ReleaseSprites
  - rtm::CoatingUnion, 58
  - rtm::ControlUnit, 62
  - rtm::CrossroadObject, 67
- Right
  - rtm, 20
- RightTurnMatrix
  - rtm::TurnObject, 92
- Rightward
  - rtm, 20
- RoadCoating
  - rtm::RoadCoating, 86, 87
- RoadType
  - rtm, 22
- Rotate\_
  - rtm::VehicleObject, 99
- Rotation\_
  - rtm::VehicleObject, 109
- RotationEnd\_
  - rtm::VehicleObject, 108
- RotationStart\_
  - rtm::VehicleObject, 107
- RotationTick\_
  - rtm::VehicleObject, 107
- RoundAngle
  - rtm, 26
- RoundCoordinate
  - rtm, 23
- RoundToCenter
  - rtm, 24
- rtm, 11
  - Allowed, 21
  - AngleToAngleType, 27
  - AngleToDirection, 28
  - AngleType, 19
  - AngleTypeToAngle, 28
  - AngleTypeToDirection, 28
  - CARS\_ACCELERATIONS, 33
  - CARS\_MAX\_SPEEDS, 32
  - COATING\_INDEXES, 31
  - CellToPixel, 27
  - CenterIsCrossed, 25
  - CheckCollisions, 23
  - Closed, 21
  - CoatingType, 21
  - CoatingUnionType, 20
  - CountDeceleration, 30
  - CrossroadSignals, 18
  - DEFAULT\_CROSSROAD\_SIGNALS, 31
  - DEFAULT\_DIRECTIONS\_SIGNAL\_SP↔
    - rites, 31
  - DirectionSignalIndex, 20
  - DirectionSignals, 18
  - DirectionToAngle, 29
  - DirectionToAngleType, 29
  - DirectionType, 20
  - Directions, 18
  - DirectionsSignalSprites, 19
  - DistanceToNextCenter, 24
  - Down, 20
  - Downward, 20
  - Forbidden, 21
  - GetFilename, 29
  - InCenter, 24
  - Left, 20
  - Leftward, 20
  - LinesCounts, 18
  - NEAR\_DELTA, 31
  - NormalizeAngle, 26
  - PixelToCell, 26
  - ROADS\_DIRECTIONS, 32
  - ROADS\_RESISTANCES, 32
  - Right, 20
  - Rightward, 20
  - RoadType, 22
  - RoundAngle, 26
  - RoundCoordinate, 23
  - RoundToCenter, 24
  - SameAngles, 25
  - SameCoordinates, 23
  - SignalFileId, 22
  - SignalSprites, 19
  - SignalType, 21
  - SignalsSprites, 19
  - Started, 21
  - StateType, 21
  - SumAngleTypes, 30
  - Up, 20
  - Upward, 20
  - Warning, 21
- rtm::BuildingObject, 36
  - BuildingObject, 37, 38
- rtm::BushObject, 38
  - BushObject, 39, 40
- rtm::CarObject, 41
  - CarObject, 43, 44
  - CheckCoatingAhead\_, 47
  - CheckCoatingUnionAhead\_, 47
  - CheckRoadAhead\_, 47
  - GetClassAcceleration\_, 48
  - GetClassMaxSpeed\_, 47
  - LineChangingStart, 45
  - MovementEnd\_, 45
  - MovementStart\_, 44
  - MovementTick\_, 45
- rtm::CoatingObject, 48
  - CoatingObject, 50
  - GetResistance, 51
  - GetSprite, 51



- HasDirection, 51
- IsDirectionAvailable, 52
- SetDirectionAvailability, 52
- SetSprite\_, 52
- SetX\_, 53
- SetY\_, 53
- rtm::CoatingUnion, 53
  - CoatingUnion, 55
  - GetCoatingObject, 56
  - GetColumn\_, 58
  - GetHeight, 56
  - GetLength, 57
  - GetRow\_, 58
  - GetType, 55
  - GetWidth, 56
  - IsCorrectColumn, 57
  - IsCorrectRow, 57
  - ReleaseSprites, 58
  - ShowSprites, 57
- rtm::ControlUnit, 59
  - ControlUnit, 60
  - GetSignal, 61
  - operator bool, 61
  - ReleaseSprites, 62
  - SetState\_, 63
  - ShowSprites, 62
  - Update, 61
  - UpdateSignal\_, 62
- rtm::CrossroadObject, 63
  - CrossroadMatrix, 65
  - CrossroadObject, 64, 65
  - GetControlUnit, 67
  - GetNullDirection, 66
  - ReleaseSprites, 67
  - ShowSprites, 67
  - TCrossroadMatrix, 66
- rtm::DrivewayObject, 68
  - CountLength\_, 72
  - CountLines\_, 73
  - DrivewayMatrix, 70
  - DrivewayObject, 69
  - GetLength, 70
  - GetLinesCount, 70
  - isLeftLine, 71, 72
  - isRightLine, 71
- rtm::DynamicObject, 73
  - CheckCollisions, 80
  - DynamicObject, 75, 76
  - GetLastDelta, 76
  - GetSpeed, 76
  - HasCollision, 77
  - IsBeholding\_, 78
  - IsIntersecting\_, 79
  - IsNear\_, 79
  - IsNearOthers, 77
  - SetCollisionFlag\_, 78
  - SetSpeed\_, 78
  - Update, 77
- rtm::MapObject, 80
  - MapObject, 81
- rtm::PuddleCoating, 83
  - PuddleCoating, 84
- rtm::RoadCoating, 85
  - GetClassDirections\_, 88
  - GetClassResistance\_, 87
  - RoadCoating, 86, 87
- rtm::SpawnType, 88
- rtm::StaticObject, 89
  - StaticObject, 90
- rtm::TurnObject, 91
  - GetAngle, 94
  - IsRight, 93
  - LeftTurnMatrix, 93
  - RightTurnMatrix, 92
  - TurnObject, 92
- rtm::VehicleObject, 94
  - AfterMoving\_, 105
  - BeforeMoving\_, 105
  - ChangeLine\_, 99
  - CheckCrossroadArea\_, 104
  - CheckForwardArea\_, 103
  - CheckForwardCoating\_, 102
  - CheckForwardCoatingUnion\_, 102
  - CheckLineChangingArea\_, 105
  - CheckMovingArea\_, 103
  - CheckRotationArea\_, 104
  - CheckTurnArea\_, 104
  - GetFinalSpeed\_, 101
  - GetMaxSpeed\_, 101
  - IsBraking\_, 100
  - IsLineChanging\_, 100
  - IsMovement\_, 100
  - IsRotation\_, 100
  - LineChanging\_, 109
  - LineChangingEnd\_, 109
  - LineChangingStart, 108
  - LineChangingTick\_, 108
  - MoveForward\_, 99
  - Movement\_, 110
  - MovementEnd\_, 107
  - MovementStart\_, 106
  - MovementTick\_, 106
  - Rotate\_, 99
  - Rotation\_, 109
  - RotationEnd\_, 108
  - RotationStart\_, 107
  - RotationTick\_, 107
  - SetBrakingFactor\_, 102
  - SetFinalSpeed\_, 101
  - SmoothBrakingCounter, 110
  - SpeedChanging\_, 110
  - Stop\_, 99
  - StopAtDistance\_, 102
  - Update, 98
  - VehicleObject, 97, 98
- rtm::WorldController, 111

- AddBuilding\_, 125
- AddBush\_, 126
- AddCar\_, 127
- AddCoatingUnion\_, 122
- AddControlUnit\_, 125
- AddCrossroad\_, 123
- AddDriveway\_, 122
- AddDynamicObject\_, 126
- AddLeftTurt\_, 124
- AddRightTurt\_, 124
- AddStaticObject\_, 125
- AddTCrossroad\_, 123
- GenerateObject\_, 121
- GetCoatingObject, 116
- GetCoatingUnion, 117
- GetColumnsCount, 115
- GetDeltaTime, 116
- GetDynamicObjects, 117
- GetLayer, 115
- GetRealColumn\_, 128
- GetRealRow\_, 128
- GetRowsCount, 116
- GetStaticObject, 117
- GetTimeFactor, 116
- GetVectorColumn\_, 127
- GetVectorRow\_, 128
- IsAllowableColumn, 118
- IsAllowableRow, 119
- IsCorrectColumn, 118
- IsCorrectRow, 118
- IsEmpty\_, 121
- IsPause, 117
- IsVisibleColumn, 119
- IsVisibleRow, 120
- LoadMap, 120, 121
- SetTimeFactor, 120
- Update, 115
- WorldController, 114, 115
- rtm::WorldObject, 129
  - GetAngle, 133
  - GetHeight, 133
  - GetSprite, 132
  - GetWidth, 133
  - GetX\_, 132
  - GetY\_, 132
  - SetAngle\_, 134
  - SetHeight\_, 135
  - SetSprite\_, 133
  - SetSpriteAngle\_, 136
  - SetSpriteHeight\_, 136
  - SetSpriteWidth\_, 136
  - SetSpriteX\_, 135
  - SetSpriteY\_, 135
  - SetWidth\_, 134
  - SetX\_, 134
  - SetY\_, 134
  - WorldObject, 131, 132
- rtm::WorldScene, 137
  - Create, 139
  - GetMainLayer, 140
  - GetMap\_, 141
  - init, 139
  - SetBackground, 140, 141
  - update, 140
- SameAngles
  - rtm, 25
- SameCoordinates
  - rtm, 23
- SetAngle\_
  - rtm::WorldObject, 134
- SetBackground
  - rtm::WorldScene, 140, 141
- SetBrakingFactor\_
  - rtm::VehicleObject, 102
- SetCollisionFlag\_
  - rtm::DynamicObject, 78
- SetDirectionAvailability
  - rtm::CoatingObject, 52
- SetFinalSpeed\_
  - rtm::VehicleObject, 101
- SetHeight\_
  - rtm::WorldObject, 135
- SetSpeed\_
  - rtm::DynamicObject, 78
- SetSprite\_
  - rtm::CoatingObject, 52
  - rtm::WorldObject, 133
- SetSpriteAngle\_
  - rtm::WorldObject, 136
- SetSpriteHeight\_
  - rtm::WorldObject, 136
- SetSpriteWidth\_
  - rtm::WorldObject, 136
- SetSpriteX\_
  - rtm::WorldObject, 135
- SetSpriteY\_
  - rtm::WorldObject, 135
- SetState\_
  - rtm::ControlUnit, 63
- SetTimeFactor
  - rtm::WorldController, 120
- SetWidth\_
  - rtm::WorldObject, 134
- SetX\_
  - rtm::CoatingObject, 53
  - rtm::WorldObject, 134
- SetY\_
  - rtm::CoatingObject, 53
  - rtm::WorldObject, 134
- ShowSprites
  - rtm::CoatingUnion, 57
  - rtm::ControlUnit, 62
  - rtm::CrossroadObject, 67
- SignalFileId
  - rtm, 22
- SignalSprites

- rtm, [19](#)
- SignalType
  - rtm, [21](#)
- SignalsSprites
  - rtm, [19](#)
- SmoothBrakingCounter
  - rtm::VehicleObject, [110](#)
- SpeedChanging\_
  - rtm::VehicleObject, [110](#)
- Started
  - rtm, [21](#)
- StateType
  - rtm, [21](#)
- StaticObject
  - rtm::StaticObject, [90](#)
- Stop\_
  - rtm::VehicleObject, [99](#)
- StopAtDistance\_
  - rtm::VehicleObject, [102](#)
- SumAngleTypes
  - rtm, [30](#)
- TCrossroadMatrix
  - rtm::CrossroadObject, [66](#)
- TurnObject
  - rtm::TurnObject, [92](#)
- Up
  - rtm, [20](#)
- Update
  - rtm::ControlUnit, [61](#)
  - rtm::DynamicObject, [77](#)
  - rtm::VehicleObject, [98](#)
  - rtm::WorldController, [115](#)
- update
  - rtm::WorldScene, [140](#)
- UpdateSignal\_
  - rtm::ControlUnit, [62](#)
- Upward
  - rtm, [20](#)
- VehicleObject
  - rtm::VehicleObject, [97](#), [98](#)
- Warning
  - rtm, [21](#)
- WorldController
  - rtm::WorldController, [114](#), [115](#)
- WorldObject
  - rtm::WorldObject, [131](#), [132](#)