

1 Readme	1
1.1 compiling	1
1.2 Developer documentation	
1.3 Wymagania do silnika 2D	1
2 Deprecated List	3
3 Namespace Index	5
3.1 Namespace List	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Class Index	9
5.1 Class List	9
6 File Index	11
6.1 File List	11
7 Namespace Documentation	13
7.1 G Namespace Reference	13
7.1.1 Variable Documentation	13
7.1.1.1 basePath	13
7.1.1.2 drwables	13
7.1.1.3 logstream	13
7.1.1.4 moveVertically	14
7.2 obstacle Namespace Reference	14
8 Class Documentation	15
8.1 AnimatedObject Class Reference	15
8.1.1 Detailed Description	16
8.1.2 Constructor & Destructor Documentation	17
8.1.2.1 ~AnimatedObject()	17
8.1.3 Member Function Documentation	17
8.1.3.1 animate()	17
8.1.3.2 getPosition()	17
8.1.3.3 move()	17
8.1.3.4 setPosition()	18
8.1.4 Member Data Documentation	19
8.1.4.1 m_position	19
8.2 AnimatedSpriteSheet Class Reference	19
8.2.1 Detailed Description	21
8.2.2 Member Typedef Documentation	21
8.2.2.1 animationData_t	21
8.2.3 Constructor & Destructor Documentation	22

22
22
22
23
23
24
25
25
25
26
27
27
28
29
29
30
30
31
31
31
31
32
32
33
33
33
33
34
34
34
34
34
35
36
36
36
36
36
37
37
38
39

8.4.3.7 setPosition()	39
8.4.3.8 setPositon()	40
8.4.3.9 update()	40
8.4.4 Member Data Documentation	41
8.4.4.1 m_dead	41
8.4.4.2 m_movementSpeed	41
8.4.4.3 m_moveVectorBase	42
8.4.4.4 m_position	42
8.4.4.5 m_timeToLife	42
8.5 Bush Class Reference	42
8.5.1 Detailed Description	44
8.5.2 Constructor & Destructor Documentation	44
8.5.2.1 Bush()	44
8.5.2.2 ~Bush()	44
8.5.3 Member Function Documentation	44
8.5.3.1 animate()	44
8.5.3.2 getCurrentSpriteRectangle()	45
8.5.3.3 getPosition()	45
8.5.3.4 move()	45
8.5.3.5 nextSprite()	46
8.5.3.6 setPosition()	47
8.5.4 Member Data Documentation	48
8.5.4.1 m_animationFrameDuration	48
8.5.4.2 m_animationTimer	48
8.5.4.3 m_aninmationFrameIndicator	48
8.5.4.4 m_position	48
8.5.4.5 s_spriteSheetPath	48
8.6 Engine Class Reference	49
8.6.1 Detailed Description	51
8.6.2 Member Typedef Documentation	51
8.6.2.1 animatedObjecsCollection_t	51
8.6.2.2 Clock	51
8.6.2.3 Drawable	51
8.6.2.4 drawableCollection_t	51
8.6.2.5 Event	52
8.6.2.6 eventHandler_t	52
8.6.2.7 RenderWindow	52
8.6.2.8 Shape	52
8.6.2.9 Time	52
8.6.3 Constructor & Destructor Documentation	52
8.6.3.1 Engine()	52
8.6.3.2 ~ Engine()	53

8.6.4 Member Function Documentation	54
8.6.4.1 add() [1/2]	54
8.6.4.2 add() [2/2]	55
8.6.4.3 animateObjects()	56
8.6.4.4 buildWindow()	56
8.6.4.5 clear()	57
8.6.4.6 display()	58
8.6.4.7 drawDrawables()	58
8.6.4.8 getInstance()	59
8.6.4.9 getLastFrameDuration()	60
8.6.4.10 getResolution()	60
8.6.4.11 getWindow()	61
8.6.4.12 handleEvents()	61
8.6.4.13 loop()	62
8.6.4.14 remove()	63
8.6.4.15 render()	64
8.6.4.16 setEventHandler()	64
8.6.4.17 setLoopFunction()	65
8.6.4.18 setMaxFps()	66
8.6.4.19 setResolution() [1/2]	66
8.6.4.20 setResolution() [2/2]	67
8.6.4.21 setWindowTitle()	68
8.6.5 Member Data Documentation	69
8.6.5.1 m_animatedObjectsCollection	69
8.6.5.2 m_clock	69
8.6.5.3 m_drawablesCollection	70
8.6.5.4 m_eventHandlers	70
8.6.5.5 m_lastFrameDuration	70
8.6.5.6 m_loopFunction	70
8.6.5.7 m_maxFPS	70
8.6.5.8 m_resoltuon	71
8.6.5.9 m_window	71
8.6.5.10 m_windowTitle	71
8.6.5.11 s_instancePtr	71
8.7 FidgetSpinner Class Reference	72
8.7.1 Detailed Description	73
8.7.2 Member Typedef Documentation	73
8.7.2.1 animationData_t	73
8.7.3 Constructor & Destructor Documentation	74
8.7.3.1 FidgetSpinner()	74
8.7.4 Member Function Documentation	74
8.7.4.1 addRotationSpeed()	74

8.7.4.2 animate()		75
8.7.4.3 getCenterPoint()		75
8.7.4.4 getCurrentAnimationFrameCount()		76
8.7.4.5 getCurrentAnimationFrameData()		76
8.7.4.6 getCurrentAnimationFrameDuration()		77
8.7.4.7 getCurrentAnimationFrameRect()		77
8.7.4.8 getPosition()		77
8.7.4.9 loadFrame()		78
8.7.4.10 loadFromConfigFile()		78
8.7.4.11 loadSpritesheet()		79
8.7.4.12 move()		80
8.7.4.13 nextFrame()	8	31
8.7.4.14 setFrame()		82
8.7.4.15 setPosition()		32
8.7.4.16 setRotationResistance()	8	33
8.7.5 Member Data Documentation		83
8.7.5.1 m_angle		83
8.7.5.2 m_animationsData		84
8.7.5.3 m_animationTimer		84
8.7.5.4 m_currentAnimationTypeIndex		34
8.7.5.5 m_currentFrameld		84
8.7.5.6 m_position		84
8.7.5.7 m_rotationResistance		35
8.7.5.8 m_rotationspeed		85
8.8 GameObject Class Reference		85
8.8.1 Detailed Description		86
8.8.2 Constructor & Destructor Documentation		86
8.8.2.1 ~GameObject()		86
8.8.3 Member Function Documentation		87
8.8.3.1 getPosition()		87
8.8.3.2 move()		87
8.8.3.3 setPosition()		87
8.8.4 Member Data Documentation		88
8.8.4.1 m_position		88
8.9 GamePlayer Class Reference		89
8.9.1 Detailed Description	9	91
8.9.2 Member Enumeration Documentation	9	91
8.9.2.1 AnimationType	9	91
8.9.2.2 MoveDirection	9	91
8.9.3 Constructor & Destructor Documentation	9	92
8.9.3.1 GamePlayer()	9	92
8.9.4 Member Function Documentation		92

8.9.4.1 animate()	92
8.9.4.2 getAnimationTypeOf()	93
8.9.4.3 getCurrentAnimationFrameSpriteRectangle()	94
8.9.4.4 getFrameCountOfAnimation()	94
8.9.4.5 getMovementSpeed()	95
8.9.4.6 getMoveVector()	95
8.9.4.7 getMoveVectorOrigin()	96
8.9.4.8 getPosition()	96
8.9.4.9 getRowOfAnimation()	96
8.9.4.10 isMoving() [1/2]	97
8.9.4.11 isMoving() [2/2]	98
8.9.4.12 isMovingDiagonaly()	98
8.9.4.13 move()	98
8.9.4.14 nextAnimationFrame()	99
8.9.4.15 setAnimationType()	00
8.9.4.16 setIsMoving()	01
8.9.4.17 setMovementSpeed()	01
8.9.4.18 setPosition()	01
8.9.4.19 stopMoving() [1/2]	02
8.9.4.20 stopMoving() [2/2]	02
8.9.4.21 update()	03
8.9.4.22 updateTextureRect()	04
8.9.5 Member Data Documentation	05
8.9.5.1 m_animationFrameDuration	05
8.9.5.2 m_animationTimer	05
8.9.5.3 m_animationType	05
8.9.5.4 m_aninmationFrameIndicator	05
8.9.5.5 m_isMoving	05
8.9.5.6 m_movementSpeed	06
8.9.5.7 m_position	06
8.9.5.8 s_spriteSheetPath	06
8.10 obstacle::LineSegment Class Reference	06
8.10.1 Detailed Description	07
8.10.2 Constructor & Destructor Documentation	07
8.10.2.1 LineSegment()	07
8.10.3 Member Function Documentation	07
8.10.3.1 draw()	07
8.10.3.2 getEnd()	80
8.10.3.3 getStart()	80
8.10.3.4 setColor()	80
8.10.3.5 setEnd()	80
8.10.3.6 setStart()	09

8.10.4 Member Data Documentation	109
8.10.4.1 m_color	109
8.10.4.2 m_points	109
8.11 Player Class Reference	110
8.11.1 Detailed Description	111
8.11.2 Member Enumeration Documentation	112
8.11.2.1 MoveDirection	112
8.11.3 Constructor & Destructor Documentation	112
8.11.3.1 ~Player()	112
8.11.4 Member Function Documentation	112
8.11.4.1 animate()	112
8.11.4.2 getMovementSpeed()	113
8.11.4.3 getMoveVector()	113
8.11.4.4 getMoveVectorOrigin()	114
8.11.4.5 getPosition()	114
8.11.4.6 isMoving() [1/2]	114
8.11.4.7 isMoving() [2/2]	115
8.11.4.8 isMovingDiagonaly()	115
8.11.4.9 move()	116
8.11.4.10 setIsMoving()	117
8.11.4.11 setMovementSpeed()	117
8.11.4.12 setPosition()	118
8.11.4.13 stopMoving() [1/2]	119
8.11.4.14 stopMoving() [2/2]	119
8.11.4.15 update()	119
8.11.5 Member Data Documentation	120
8.11.5.1 m_isMoving	120
8.11.5.2 m_movementSpeed	120
8.11.5.3 m_position	121
8.12 Point2d Class Reference	121
8.12.1 Detailed Description	122
8.12.2 Member Typedef Documentation	122
8.12.2.1 cordinate_t	122
8.12.3 Constructor & Destructor Documentation	122
8.12.3.1 Point2d() [1/4]	122
8.12.3.2 Point2d() [2/4]	123
8.12.3.3 Point2d() [3/4]	123
8.12.3.4 Point2d() [4/4]	123
8.12.4 Member Function Documentation	123
8.12.4.1 distanceTo()	123
8.12.4.2 getX()	124
8.12.4.3 getY()	124

	8.12.4.4 length()	125
	8.12.4.5 setX()	125
	8.12.4.6 setY()	126
	8.12.4.7 swap()	126
	8.12.4.8 to Vector2f()	127
	8.12.5 Friends And Related Symbol Documentation	127
	8.12.5.1 operator"!=	127
	8.12.5.2 operator* [1/2]	128
	8.12.5.3 operator* [2/2]	128
	8.12.5.4 operator+	128
	8.12.5.5 operator+=	128
	8.12.5.6 operator	128
	8.12.5.7 operator <<	129
	8.12.5.8 operator==	129
	8.12.5.9 operator>>	129
	8.12.6 Member Data Documentation	129
	8.12.6.1 m_x	129
	8.12.6.2 m_y	129
	8.13 UpdateableObject Class Reference	130
	8.13.1 Detailed Description	130
	8.13.2 Member Function Documentation	130
	8.13.2.1 update()	130
		404
9 1		131
	9.1 README.md File Reference	
	9.2 animatedObject.cpp File Reference	
	9.3 animatedObject.cpp	
		132
	9.5 animatedObject.hpp	
	9.6 animatedSpriteSheet.cpp File Reference	
	9.6.1 Function Documentation	
	9.6.1.1 operator>>()	
		133
	9.8 animatedSpriteSheet.hpp File Reference	
	·	136
	3 11	137
	9.11 engine.cpp	137
		139
	9.13 engine.hpp	139 140
	9.13 engine.hpp	139 140 141
	9.13 engine.hpp	139 140 141 142

9.17 GameObject.hpp
9.18 lineSegment.cpp File Reference
9.18.1 Detailed Description
9.19 lineSegment.cpp
9.20 lineSegment.hpp File Reference
9.20.1 Detailed Description
9.21 lineSegment.hpp
9.22 log.cpp File Reference
9.23 log.cpp
9.24 log.hpp File Reference
9.24.1 Macro Definition Documentation
9.24.1.1 LOGERROR
9.24.1.2 LOGINFO
9.24.1.3 LOGINFON
9.24.1.4 LOGTRACEN
9.24.1.5 LOGWARN
9.24.1.6 LOGWARNN
9.25 log.hpp
9.26 player.cpp File Reference
9.27 player.cpp
9.28 player.hpp File Reference
9.29 player.hpp
9.30 point2d.cpp File Reference
9.30.1 Function Documentation
9.30.1.1 operator"!=()
9.30.1.2 operator*() [1/2]
9.30.1.3 operator*() [2/2]
9.30.1.4 operator+()
9.30.1.5 operator+=()
9.30.1.6 operator-()
9.30.1.7 operator<<()
9.30.1.8 operator==()
9.30.1.9 operator>>()
9.31 point2d.cpp
9.32 point2d.hpp File Reference
9.33 point2d.hpp
9.34 updateableObject.cpp File Reference
9.35 updateableObject.cpp
9.36 updateableObject.hpp File Reference
9.37 updateableObject.hpp
9.38 test/main.cpp File Reference
9.38.1 Function Documentation

9.38.1.1 addSomeRenderables()
9.38.1.2 customLoopFunction()
9.38.1.3 main()
9.38.1.4 setUpCustomEvents()
9.39 test/main.cpp
9.40 test2/main.cpp File Reference
9.40.1 Function Documentation
9.40.1.1 customLoop()
9.40.1.2 getRandomColor()
9.40.1.3 handleBalls()
9.40.1.4 handlePlayerMovement()
9.40.1.5 initializeBush()
9.40.1.6 initializeFidgetSpinner()
9.40.1.7 initialziePlayer()
9.40.1.8 keyPressedEventHandler()
9.40.1.9 keyReleasedEventHandler()
9.40.1.10 main()
9.40.1.11 spinTheFidget()
9.40.1.12 throwBall()
9.40.1.13 throwBallPlayer()
9.40.2 Variable Documentation
9.40.2.1 balls
9.40.2.2 bush
9.40.2.3 eng
9.40.2.4 fidgetSpinner
9.40.2.5 player
9.41 test2/main.cpp
9.42 test3/main.cpp File Reference
9.42.1 Function Documentation
9.42.1.1 main()
9.42.2 Variable Documentation
9.42.2.1 animation
9.43 test3/main.cpp
9.44 ball.cpp File Reference
9.45 ball.cpp
9.46 ball.hpp File Reference
9.47 ball.hpp
9.48 bush.cpp File Reference
9.49 bush.cpp
9.50 bush.hpp File Reference
9.51 bush.hpp
9.52 fidgetSpinner.cpp File Reference

Inc	lex	191
	9.59 gamePlayer.hpp	189
	9.58 gamePlayer.hpp File Reference	188
	9.57 gamePlayer.cpp	186
	9.56 gamePlayer.cpp File Reference	186
	9.55 fidgetSpinner.hpp	185
	9.54 fidgetSpinner.hpp File Reference	185
	9.53 fidgetSpinner.cpp	184

Readme

1.1 compiling

```
mkdir build
cd build
cmake ..
# replace 4 with number of cores for compiling
make -i 4
```

1.2 Developer documentation

• During compilation doxygen generates in doc_doxygen subdirectory in cmake build directory.

1.3 Wymagania do silnika 2D

- · Obsługa klawiatury i myszy
- · Obsługa współrzędnych (Point2D)
- · Rysowanie prymitywów
- Wypełnianie prymitywów kolorem
- · Przekształcenia geometryczne
- Hierarchia klas
- · Obsługa bitmap
- · Animowanie bitmap
- · Demo technologiczne (do obrony)
- Sprawozdanie i dokumentacja

2 Readme

Deprecated List

File lineSegment.cpp

Bad design. Reinventing wheel.

File lineSegment.hpp

Bad design. Reinventing wheel.

Deprecated List

Namespace Index

Here is a list of all namespaces with brief descriptions:

3.1 Namespace List

G.							 																	13	3
obstac	cle				 		 																	14	4

6 Namespace Index

Hierarchical Index

4.1 Class Hierarchy

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

AnimatedSpriteSheet::AnimationFrameData sf::CircleShape		 											3
•													_
Ball													
Engine													
GameObject		 					 						8
AnimatedObject		 									 		. 1
AnimatedSpriteSheet	 	 			 						 		. 1
FidgetSpinner	 												. 7
Bush	 	 			 						 		. 4
Player	 	 			 						 		. 11
GamePlayer	 												. 8
Ball		 									 		. 3
obstacle::LineSegment		 					 						10
Point2d		 					 						12
sf::Sprite													
AnimatedObject		 									 		. 1
sf::Texture													
AnimatedObject		 									 		. 1
UpdateableObject		 					 						13
Ball		 									 		. 3
Player		 									 		.11

8 **Hierarchical Index**

Class Index

5.1 Class List

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

AnimatedObject
Animated object class
AnimatedSpriteSheet
AnimatedSpriteSheet::AnimationFrameData
Ball 34
Bush 42
Engine
The engine class
FidgetSpinner
GameObject
Game object class
GamePlayer
obstacle::LineSegment
Player
Player class
Point2d
Class containing 2D point co-ordinates
UpdateableObject
Updateable object class

10 Class Index

File Index

6.1 File List

Here is a list of all files with brief descriptions:

animatedObject.cpp	
animatedObject.hpp	
animatedSpriteSheet.cpp	
animatedSpriteSheet.hpp	
engine.cpp	
engine.hpp	
GameObject.cpp	
GameObject.hpp	
lineSegment.cpp	
lineSegment.hpp	
log.cpp	
log.hpp	
player.cpp	
player.hpp	
point2d.cpp	
point2d.hpp	
updateableObject.cpp	
updateableObject.hpp	
test/main.cpp	
test2/main.cpp	
test3/main.cpp	
ball.cpp	
ball.hpp	
bush.cpp	
bush.hpp	
fidgetSpinner.cpp	
fidgetSpinner.hpp	
gamePlayer.cpp	
gamePlayer.hpp	

12 File Index

Namespace Documentation

7.1 G Namespace Reference

Variables

- std::ostream & logstream {std::cerr}
- std::vector< Engine::Shape * > drwables {}
- bool moveVertically {false}
- std::string basePath = "resources/"

7.1.1 Variable Documentation

7.1.1.1 basePath

```
std::string G::basePath = "resources/"
```

Definition at line 7 of file test3/main.cpp.

7.1.1.2 drwables

```
std::vector<Engine::Shape *> G::drwables {}
```

Definition at line 12 of file test/main.cpp.

Referenced by addSomeRenderables(), and customLoopFunction().

7.1.1.3 logstream

```
std::ostream & G::logstream {std::cerr}

Definition at line 7 of file log.cpp.
00007 {std::cerr};
```

7.1.1.4 moveVertically

```
bool G::moveVertically {false}

Definition at line 13 of file test/main.cpp.
00013 {false};
```

Referenced by customLoopFunction(), and setUpCustomEvents().

7.2 obstacle Namespace Reference

Classes

• class LineSegment

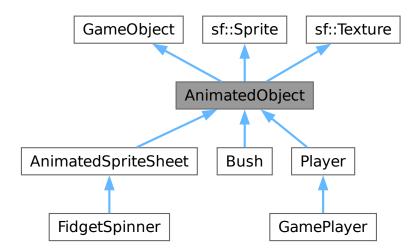
Class Documentation

8.1 AnimatedObject Class Reference

Animated object class.

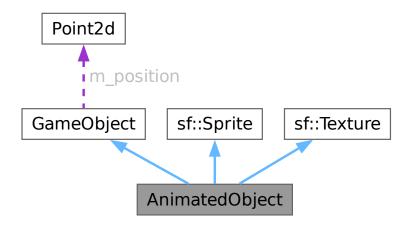
#include <animatedObject.hpp>

Inheritance diagram for AnimatedObject:



16 Class Documentation

Collaboration diagram for AnimatedObject:



Public Member Functions

virtual ∼AnimatedObject ()

Destructor.

virtual void setPosition (Point2d pos)

Object position setter.

• virtual void animate ()

Animates object.

• virtual void move (Point2d vec)

Translates object in space.

• virtual Point2d getPosition () const

Object position getter.

Private Attributes

Point2d m_position {}

Position of object on screen.

8.1.1 Detailed Description

Animated object class.

Definition at line 13 of file animatedObject.hpp.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 ~AnimatedObject()

```
virtual AnimatedObject::~AnimatedObject ( ) [inline], [virtual]
```

Destructor.

Definition at line 21 of file animatedObject.hpp.

8.1.3 Member Function Documentation

8.1.3.1 animate()

```
void AnimatedObject::animate ( ) [virtual]
```

Animates object.

Reimplemented in Bush, AnimatedSpriteSheet, FidgetSpinner, and GamePlayer.

```
Definition at line 5 of file animatedObject.cpp.
```

References LOGWARN.

Referenced by Engine::animateObjects().

Here is the caller graph for this function:

```
Engine::render Engine::animateObjects AnimatedObject::animate
```

8.1.3.2 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

```
Definition at line 7 of file GameObject.cpp.
00007 { return m_position; }
```

References GameObject::m_position.

8.1.3.3 move()

Translates object in space.

18 Class Documentation

Parameters

vector 2D vector added to current object position.

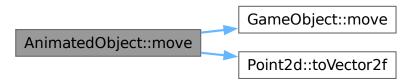
Reimplemented from GameObject.

Reimplemented in Player.

Definition at line 14 of file animatedObject.cpp.

References GameObject::move(), and Point2d::toVector2f().

Here is the call graph for this function:



8.1.3.4 setPosition()

Object position setter.

Parameters

```
pos Position to be set.
```

Reimplemented from GameObject.

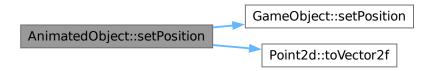
Definition at line 9 of file animatedObject.cpp.

```
00009
00010    GameObject::setPosition(pos);
00011    sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

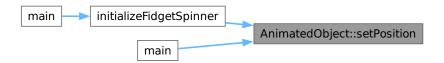
References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.4 Member Data Documentation

8.1.4.1 m_position

Point2d GameObject::m_position {} [private], [inherited]

Position of object on screen.

Definition at line 14 of file GameObject.hpp. 00014 {};

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

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

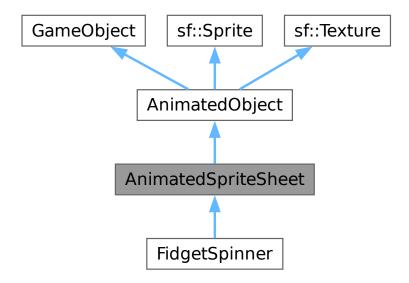
- · animatedObject.hpp
- animatedObject.cpp

8.2 AnimatedSpriteSheet Class Reference

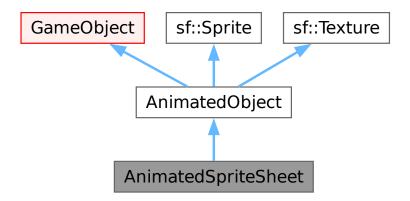
#include <animatedSpriteSheet.hpp>

20 Class Documentation

Inheritance diagram for AnimatedSpriteSheet:



Collaboration diagram for AnimatedSpriteSheet:



Classes

• struct AnimationFrameData

Public Types

using animationData_t = std::vector< AnimationFrameData >

Public Member Functions

- AnimatedSpriteSheet (std::string_view path)
- virtual void animate () override

Animates object.

• virtual void setPosition (Point2d pos)

Object position setter.

virtual void move (Point2d vec)

Translates object in space.

· virtual Point2d getPosition () const

Object position getter.

Protected Member Functions

- int getCurrentAnimationFrameCount () const
- sf::IntRect getCurrentAnimationFrameRect () const
- float getCurrentAnimationFrameDuration () const
- const AnimationFrameData & getCurrentAnimationFrameData () const
- void nextFrame ()
- void setFrame (const AnimationFrameData &frameData)
- void loadFrame (std::istream &stream, animationData_t &animationData)

load frame data from stream

- void loadSpritesheet (std::istream &stream, std::string_view configFileDirectoryPath)
- void loadFromConfigFile (std::string_view pathToDir)

Private Attributes

- std::vector< animationData_t > m_animationsData
- int m_currentAnimationTypeIndex
- int m currentFrameId {}
- float m_animationTimer {}
- Point2d m_position {}

Position of object on screen.

8.2.1 Detailed Description

Definition at line 8 of file animatedSpriteSheet.hpp.

8.2.2 Member Typedef Documentation

8.2.2.1 animationData_t

using AnimatedSpriteSheet::animationData_t = std::vector<AnimationFrameData>

Definition at line 24 of file animatedSpriteSheet.hpp.

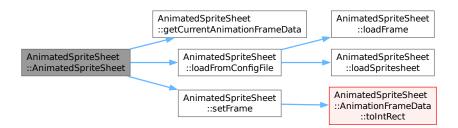
22 Class Documentation

8.2.3 Constructor & Destructor Documentation

8.2.3.1 AnimatedSpriteSheet()

References getCurrentAnimationFrameData(), loadFromConfigFile(), LOGINFO, and setFrame().

Here is the call graph for this function:



8.2.4 Member Function Documentation

8.2.4.1 animate()

```
void AnimatedSpriteSheet::animate ( ) [override], [virtual]
```

Animates object.

Reimplemented from AnimatedObject.

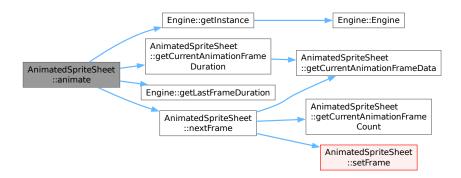
Reimplemented in FidgetSpinner.

Definition at line 37 of file animatedSpriteSheet.cpp.

```
00037
00038 LOGTRACEN;
00039 m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
00040 // if duration of frame not exhaused do nothing;
00041 if (m_animationTimer < getCurrentAnimationFrameDuration())
00042 return;
00043 nextFrame();
00044 }</pre>
```

References getCurrentAnimationFrameDuration(), Engine::getInstance(), Engine::getLastFrameDuration(), LOGTRACEN, m animationTimer, and nextFrame().

Here is the call graph for this function:



8.2.4.2 getCurrentAnimationFrameCount()

int AnimatedSpriteSheet::getCurrentAnimationFrameCount () const [protected]

```
Definition at line 54 of file animatedSpriteSheet.cpp.

00054

00055

LOGTRACEN;

00056

return m_animationsData[m_currentAnimationTypeIndex].size();

00057
```

References LOGTRACEN, m animationsData, and m currentAnimationTypeIndex.

Referenced by nextFrame().

Here is the caller graph for this function:



8.2.4.3 getCurrentAnimationFrameData()

const AnimatedSpriteSheet::AnimationFrameData & AnimatedSpriteSheet::getCurrentAnimation←
FrameData () const [protected]

Definition at line 60 of file animatedSpriteSheet.cpp.

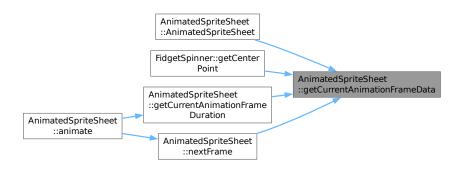
```
00060
        LOGTRACEN:
00061
        if (m_animationsData.size() == 0 ||
00062
            m_animationsData.size() < m_currentAnimationTypeIndex) {</pre>
00063
00064
          LOGWARN « "No animation data\n";
00065
        } else if (m_animationsData[m_currentAnimationTypeIndex].size() == 0 ||
00066
                    m_animationsData[m_currentAnimationTypeIndex].size() <</pre>
          m_currentAnimationTypeIndex) {
LOGWARN « "No frame data in animation " « m_currentAnimationTypeIndex
00067
00068
00069
00070
        return m_animationsData[m_currentAnimationTypeIndex][m_currentFrameId];
```

```
00072 }
```

References LOGTRACEN, LOGWARN, m animationsData, m currentAnimationTypeIndex, and m currentFrameId.

Referenced by AnimatedSpriteSheet(), FidgetSpinner::getCenterPoint(), getCurrentAnimationFrameDuration(), and nextFrame().

Here is the caller graph for this function:



8.2.4.4 getCurrentAnimationFrameDuration()

float AnimatedSpriteSheet::getCurrentAnimationFrameDuration () const [protected]

```
Definition at line 74 of file animatedSpriteSheet.cpp. 00074
```

```
00074
00075 LOGTRACEN;
00076 return getCurrentAnimationFrameData().m_duration;
00077 };
```

 $References\ getCurrent Animation Frame Data(), LOGTRACEN, and Animated SpriteSheet:: Animation Frame Data:: m_duration.$

{

Referenced by animate().

Here is the call graph for this function:





8.2.4.5 getCurrentAnimationFrameRect()

sf::IntRect AnimatedSpriteSheet::getCurrentAnimationFrameRect () const [protected]

8.2.4.6 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

Definition at line 7 of file GameObject.cpp. 00007 { return m_position; }

References GameObject::m_position.

8.2.4.7 loadFrame()

load frame data from stream

Definition at line 85 of file animatedSpriteSheet.cpp.

```
00087
00088
        // load Frame data
00089
        AnimationFrameData fdat{};
00090
        stream » fdat;
// LOGINFO « fdat.m_position « " " « fdat.m_size « " " «
00091
00092
        // fdat.m_duration
00093
                    « '\n';
00094
00095
        \ensuremath{//} add frame data to animation
00096
        animationData.push_back(fdat);
        stream.ignore(std::numeric\_limits < std::streamsize > :: max(), ~ ' \n');
00097
00098 };
```

References LOGTRACEN.

Referenced by loadFromConfigFile().



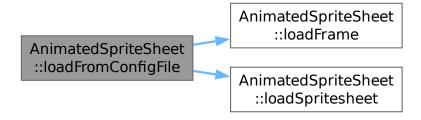
8.2.4.8 loadFromConfigFile()

```
void AnimatedSpriteSheet::loadFromConfigFile (
               std::string_view pathToDir ) [protected]
Definition at line 113 of file animatedSpriteSheet.cpp.
00114
        std::string configFilePath{static_cast<std::string>(pathToDir) +
00115
                                      "/config.txt"};
00116
        // open file
00117
        std::fstream configFile{configFilePath};
        if (!configFile.is_open()) {
   LOGERROR « "opening config file failed it should be at " « configFilePath
00118
00119
                   « "\n";
00120
00121
          return;
00122
00123
00124
        animationData_t *animationDataBeingLoaded{NULL};
00125
        std::string line{};
00126
00127
        while (!configFile.eof()) {
         line = "";
00128
00129
          // load line, ignore lines that are commnets
00130
00131
            std::getline(configFile, line);
00132
          } while (line.starts_with("#"));
00133
00134
          // load spritesheet
00135
          if (line.starts_with("SPRITESHEET")) {
00136
             loadSpritesheet(configFile, pathToDir);
          } else if (line.starts_with("ANIMATION"))
00137
00138
            // Add new animation and change pointer
                   LOGINFO « "loading animation data\n";
00139
            m_animationsData.push_back({});
00140
            animationDataBeingLoaded = {&m_animationsData.back()};
00142
00143
          // load frame
          else if (line.starts_with("FRAME"))
00144
            if (animationDataBeingLoaded == NULL)
00145
              LOGERROR « "config file is corrupted, FRAME before ANIMATION?";
00146
00147
            loadFrame(configFile, *animationDataBeingLoaded);
00148
00149
          // not known
00150
          else {
            // empty line ignore
if (line == "") {
00151
00152
00153
              continue;
00154
00155
             // invalid line
            LOGWARN « "not recegnized config option: " « line « " in file: " « configFilePath « '\n';
00156
00157
00158
          }
00159
00160
        // close file
00161
        configFile.close();
```

References loadFrame(), loadSpritesheet(), LOGERROR, LOGWARN, and m_animationsData.

Referenced by AnimatedSpriteSheet().

00162 }



Here is the caller graph for this function:



8.2.4.9 loadSpritesheet()

Definition at line 100 of file animatedSpriteSheet.cpp.

References LOGINFO.

Referenced by loadFromConfigFile().

Here is the caller graph for this function:



8.2.4.10 move()

Translates object in space.

Parameters

vector | 2D vector added to current object position.

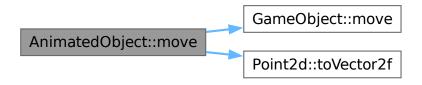
Reimplemented from GameObject.

Reimplemented in Player.

Definition at line 14 of file animatedObject.cpp.

References GameObject::move(), and Point2d::toVector2f().

Here is the call graph for this function:



8.2.4.11 nextFrame()

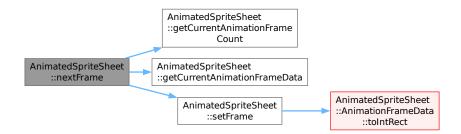
```
void AnimatedSpriteSheet::nextFrame ( ) [protected]
```

Definition at line 46 of file animatedSpriteSheet.cpp.

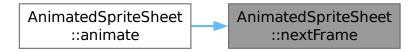
```
00046
00047 LOGTRACEN;
00048 // change frame
00049 ++m_currentFrameId;
00050 m_currentFrameId %= getCurrentAnimationFrameCount();
00051 setFrame(getCurrentAnimationFrameData());
00052 }
```

References getCurrentAnimationFrameCount(), getCurrentAnimationFrameData(), LOGTRACEN, m_currentFrameId, and setFrame().

Referenced by animate().



Here is the caller graph for this function:



8.2.4.12 setFrame()

References LOGTRACEN, m_animationTimer, and AnimatedSpriteSheet::AnimationFrameData::toIntRect().

Referenced by AnimatedSpriteSheet(), and nextFrame().

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.4.13 setPosition()

Object position setter.

Parameters

```
pos Position to be set.
```

Reimplemented from GameObject.

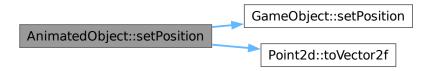
Definition at line 9 of file animatedObject.cpp.

```
00009
00010   GameObject::setPosition(pos);
00011   sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

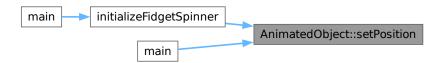
References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.5 Member Data Documentation

8.2.5.1 m_animationsData

 $\verb|std::vector<| animationData_t>| AnimatedSpriteSheet::m_animationsData|| [private]||$

Colecction of animations.

Definition at line 55 of file animatedSpriteSheet.hpp.

Referenced by getCurrentAnimationFrameCount(), getCurrentAnimationFrameData(), and loadFromConfigFile().

8.2.5.2 m_animationTimer

```
float AnimatedSpriteSheet::m_animationTimer {} [private]
```

Definition at line 63 of file animatedSpriteSheet.hpp.

Referenced by animate(), and setFrame().

8.2.5.3 m_currentAnimationTypeIndex

```
int AnimatedSpriteSheet::m_currentAnimationTypeIndex [private]
```

Indicates index of current animation in m animationsData.

Definition at line 60 of file animatedSpriteSheet.hpp.

Referenced by getCurrentAnimationFrameCount(), and getCurrentAnimationFrameData().

8.2.5.4 m_currentFrameId

```
int AnimatedSpriteSheet::m_currentFrameId {} [private]
```

Definition at line 61 of file animatedSpriteSheet.hpp.

Referenced by getCurrentAnimationFrameData(), and nextFrame().

8.2.5.5 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

Definition at line 14 of file GameObject.hpp. 00014 {};

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

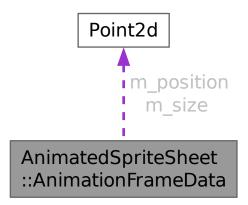
The documentation for this class was generated from the following files:

- · animatedSpriteSheet.hpp
- · animatedSpriteSheet.cpp

8.3 AnimatedSpriteSheet::AnimationFrameData Struct Reference

#include <animatedSpriteSheet.hpp>

Collaboration diagram for AnimatedSpriteSheet::AnimationFrameData:



Public Member Functions

• sf::IntRect toIntRect () const

Public Attributes

- · float m duration
- Point2d m_size
- Point2d m_position

Friends

• std::istream & operator>> (std::istream &is, AnimationFrameData &afd)

8.3.1 Detailed Description

Definition at line 11 of file animatedSpriteSheet.hpp.

8.3.2 Member Function Documentation

8.3.2.1 toIntRect()

sf::IntRect AnimatedSpriteSheet::AnimationFrameData::toIntRect () const

Definition at line 20 of file animatedSpriteSheet.cpp.

```
00020
00021 LOGTRACEN;
00022 return {static_cast<int>(m_position.getX()),
00023 static_cast<int>(m_position.getY()), static_cast<int>(m_size.getX()),
00024 static_cast<int>(m_size.getY())};
```

References Point2d::getX(), Point2d::getY(), LOGTRACEN, m_position, and m_size.

Referenced by AnimatedSpriteSheet::setFrame().

Here is the call graph for this function:



Here is the caller graph for this function:



8.3.3 Friends And Related Symbol Documentation

8.3.3.1 operator>>

8.3.4 Member Data Documentation

8.3.4.1 m duration

 $\verb|float AnimatedSpriteSheet::AnimationFrameData::m_duration|\\$

Definition at line 13 of file animatedSpriteSheet.hpp.

Referenced by AnimatedSpriteSheet::getCurrentAnimationFrameDuration().

8.3.4.2 m_position

Point2d AnimatedSpriteSheet::AnimationFrameData::m_position

Definition at line 17 of file animatedSpriteSheet.hpp.

Referenced by FidgetSpinner::getCenterPoint(), and toIntRect().

8.3.4.3 m_size

Point2d AnimatedSpriteSheet::AnimationFrameData::m_size

Definition at line 15 of file animatedSpriteSheet.hpp.

Referenced by FidgetSpinner::getCenterPoint(), and toIntRect().

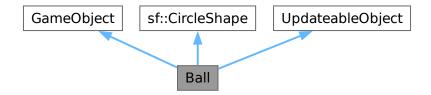
The documentation for this struct was generated from the following files:

- animatedSpriteSheet.hpp
- animatedSpriteSheet.cpp

8.4 Ball Class Reference

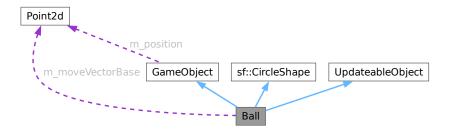
#include <ball.hpp>

Inheritance diagram for Ball:



8.4 Ball Class Reference 35

Collaboration diagram for Ball:



Public Member Functions

- Ball (float radius=10, std::size_t pointCount=30)
- void setMovementSpeed (float movementSpeed)
- void setMoveVectorBase (Point2d moveVectorOrigin)
- · virtual void update ()

Updates state of object.

virtual void move (Point2d vector)

Translates object in space.

- virtual void setPositon (Point2d pos)
- bool isDead () const
- virtual void setPosition (Point2d pos)

Object position setter.

• virtual Point2d getPosition () const

Object position getter.

Private Member Functions

• Point2d getMoveVector ()

Private Attributes

- float m_movementSpeed {10}
- Point2d m_moveVectorBase {}
- float m_timeToLife {15}
- bool m_dead {false}
- Point2d m_position {}

Position of object on screen.

8.4.1 Detailed Description

Definition at line 10 of file ball.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Ball()

8.4.3 Member Function Documentation

8.4.3.1 getMoveVector()

```
Point2d Ball::getMoveVector ( ) [private]

Definition at line 43 of file ball.cpp.
00043 { return {m_moveVectorBase * m_movementSpeed}; }
```

References m_movementSpeed, and m_moveVectorBase.

Referenced by update().

Here is the caller graph for this function:



8.4.3.2 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

Definition at line 7 of file GameObject.cpp.
00007 { return m_position; }

References GameObject::m_position.

8.4 Ball Class Reference 37

8.4.3.3 isDead()

```
Definition at line 30 of file ball.cpp.
00030 { return m_dead; }
```

References m_dead.

Referenced by handleBalls().

Here is the caller graph for this function:



8.4.3.4 move()

Translates object in space.

Parameters

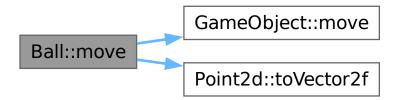
vector | 2D vector added to current object position.

Reimplemented from GameObject.

References GameObject::move(), and Point2d::toVector2f().

Referenced by update().

Here is the call graph for this function:



Here is the caller graph for this function:



8.4.3.5 setMovementSpeed()

References m_movementSpeed.

Referenced by throwBallPlayer().



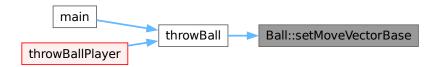
8.4 Ball Class Reference 39

8.4.3.6 setMoveVectorBase()

References m_moveVectorBase.

Referenced by throwBall().

Here is the caller graph for this function:



8.4.3.7 setPosition()

Object position setter.

Parameters

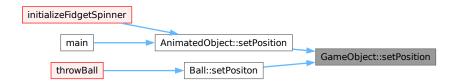


Reimplemented in AnimatedObject.

Definition at line 5 of file GameObject.cpp. 00005 { m_position = pos; }

References GameObject::m_position.

Referenced by AnimatedObject::setPosition(), and setPositon().

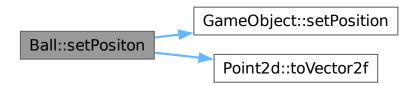


8.4.3.8 setPositon()

References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by throwBall().

Here is the call graph for this function:



Here is the caller graph for this function:



8.4.3.9 update()

```
void Ball::update ( ) [virtual]
```

Updates state of object.

Implements UpdateableObject.

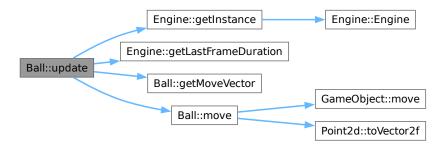
```
Definition at line 32 of file ball.cpp.
```

8.4 Ball Class Reference 41

References Engine::getInstance(), Engine::getLastFrameDuration(), getMoveVector(), m_dead, m_timeToLife, and move().

Referenced by handleBalls().

Here is the call graph for this function:



Here is the caller graph for this function:



8.4.4 Member Data Documentation

8.4.4.1 m_dead

```
bool Ball::m_dead {false} [private]

Definition at line 32 of file ball.hpp.
00032 {false};
```

Referenced by isDead(), and update().

8.4.4.2 m_movementSpeed

```
float Ball::m_movementSpeed {10} [private]

Definition at line 29 of file ball.hpp.
00029 {10};
```

Referenced by getMoveVector(), and setMovementSpeed().

8.4.4.3 m_moveVectorBase

```
Point2d Ball::m_moveVectorBase {} [private]
```

Definition at line 30 of file ball.hpp.

Referenced by getMoveVector(), and setMoveVectorBase().

8.4.4.4 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

Definition at line 14 of file GameObject.hpp. 00014 {};

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

8.4.4.5 m timeToLife

```
float Ball::m_timeToLife {15} [private]
```

Definition at line 31 of file ball.hpp. 00031 {15};

Referenced by update().

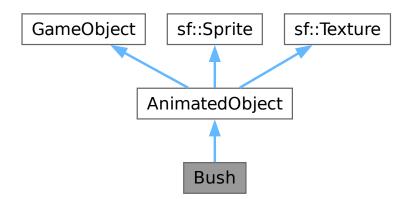
The documentation for this class was generated from the following files:

- ball.hpp
- ball.cpp

8.5 Bush Class Reference

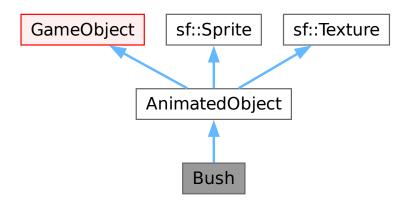
```
#include <bush.hpp>
```

Inheritance diagram for Bush:



8.5 Bush Class Reference 43

Collaboration diagram for Bush:



Public Member Functions

- Bush ()
- ∼Bush ()
- virtual void animate ()

Animates object.

• virtual void setPosition (Point2d pos)

Object position setter.

virtual void move (Point2d vec)

Translates object in space.

· virtual Point2d getPosition () const

Object position getter.

Private Member Functions

- void nextSprite ()
- sf::IntRect getCurrentSpriteRectangle (int frameNumber)

Private Attributes

- int m_aninmationFrameIndicator {}
- float m_animationFrameDuration {1}
- float m_animationTimer {}
- Point2d m_position {}

Position of object on screen.

Static Private Attributes

• static constexpr std::string_view s_spriteSheetPath {"resource/bush/bush.png"}

8.5.1 Detailed Description

Definition at line 8 of file bush.hpp.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Bush()

```
Bush::Bush ( )
```

Definition at line 10 of file bush.cpp.

```
00010 {
00011 LOGINFON;
00012 loadFromFile(std::string(s_spriteSheetPath), sf::IntRect{0, 0, 300, 150});
00013 setTexture(*this);
00014 setTextureRect(getCurrentSpriteRectangle(0));
00015 }
```

References getCurrentSpriteRectangle(), LOGINFON, and s_spriteSheetPath.

Here is the call graph for this function:



8.5.2.2 ∼Bush()

```
Bush::\simBush ( )
```

Definition at line 8 of file bush.cpp.

8.5.3 Member Function Documentation

8.5.3.1 animate()

```
void Bush::animate ( ) [virtual]
```

Animates object.

Reimplemented from AnimatedObject.

```
Definition at line 17 of file bush.cpp.
```

```
00017 {
00018 m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
```

8.5 Bush Class Reference 45

```
00019    if (m_animationTimer < m_animationFrameDuration)
00020         return;
00021
00022    m_animationTimer = 0;
00023    nextSprite();
00024 }</pre>
```

References Engine::getInstance(), Engine::getLastFrameDuration(), m_animationFrameDuration, m_animationTimer, and nextSprite().

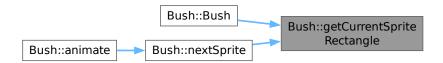
Here is the call graph for this function:



8.5.3.2 getCurrentSpriteRectangle()

Referenced by Bush(), and nextSprite().

Here is the caller graph for this function:



8.5.3.3 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

Definition at line 7 of file GameObject.cpp.

```
00007 { return m_position; }
```

References GameObject::m_position.

8.5.3.4 move()

Translates object in space.

Parameters

vector	2D vector added to current object position.
--------	---

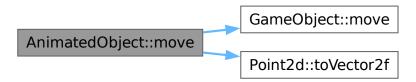
Reimplemented from GameObject.

Reimplemented in Player.

```
Definition at line 14 of file animatedObject.cpp.
```

References GameObject::move(), and Point2d::toVector2f().

Here is the call graph for this function:



8.5.3.5 nextSprite()

```
void Bush::nextSprite ( ) [private]
```

Definition at line 26 of file bush.cpp.

```
00026
00027
this->setTextureRect(getCurrentSpriteRectangle(m_aninmationFrameIndicator));
00028
++m_aninmationFrameIndicator;
00029
m_aninmationFrameIndicator %= 2;
00030
}
```

 $References\ getCurrentSpriteRectangle(),\ and\ m_aninmationFrameIndicator.$

Referenced by animate().



8.5 Bush Class Reference 47

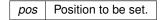
Here is the caller graph for this function:



8.5.3.6 setPosition()

Object position setter.

Parameters



Reimplemented from GameObject.

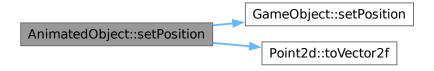
Definition at line 9 of file animatedObject.cpp.

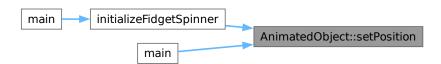
```
00009
00010    GameObject::setPosition(pos);
00011    sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:





8.5.4 Member Data Documentation

8.5.4.1 m_animationFrameDuration

```
float Bush::m_animationFrameDuration {1} [private]

Definition at line 11 of file bush.hpp.
00011 {1};
```

Referenced by animate().

8.5.4.2 m animationTimer

```
float Bush::m_animationTimer {} [private]

Definition at line 12 of file bush.hpp.
00012 {}:
```

Referenced by animate().

8.5.4.3 m_aninmationFrameIndicator

```
int Bush::m_aninmationFrameIndicator {} [private]

Definition at line 10 of file bush.hpp.
00010 {};
```

Referenced by nextSprite().

8.5.4.4 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

```
Definition at line 14 of file GameObject.hpp.
```

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

8.5.4.5 s spriteSheetPath

```
constexpr std::string_view Bush::s_spriteSheetPath {"resource/bush/bush.png"} [static], [constexpr],
[private]
```

Definition at line 9 of file bush.hpp. 00009 {"resource/bush/bush.png"};

Referenced by Bush().

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

- bush.hpp
- bush.cpp

8.6 Engine Class Reference

The engine class.

```
#include <engine.hpp>
```

Collaboration diagram for Engine:



Public Types

- using Time = sf::Time
- using Clock = sf::Clock
- using RenderWindow = sf::RenderWindow
- using Event = sf::Event
- using Drawable = sf::Drawable
- using Shape = sf::Shape
- using eventHandler_t = std::function < void(const Event &)>
- using drawableCollection t = std::set< Drawable * >
- using animatedObjecsCollection_t = std::set< AnimatedObject * >

Public Member Functions

• Engine ()

Constructor.

∼Engine ()

Destructor.

Engine & setMaxFps (int fps)

Set Max FPS.

Engine & setWindowTitle (std::string_view title)

Set window title.

• Engine & setResolution (Point2d resolution)

Set resolution.

Engine & setResolution (Point2d::cordinate t x, Point2d::cordinate t y)

Set resolution.

Engine & setLoopFunction (std::function< void()> function)

Set function that is invoked in the main loop.

• Engine & setEventHandler (Event::EventType eventType, eventHandler_t handler)

Set custom event handler.

· Point2d getResolution () const

Resolution getter.

• Engine & buildWindow ()

Builds the window.

void handleEvents ()

Handles events.

• void clear ()

Clears window to a single color.

· void render ()

Renders objects.

• void display ()

Displays rendered objects.

• void loop ()

The main loop.

• RenderWindow & getWindow ()

Window getter.

• void add (Drawable *drawable)

Adds drawable to Collection.

void add (AnimatedObject *animatedObject)

Adds animated object to Collection.

- void remove (Drawable *)
- Time getLastFrameDuration () const

Last frame duration getter.

Static Public Member Functions

static Engine & getInstance ()
 Instance getter.

Private Member Functions

• void drawDrawables ()

Draws objects.

void animateObjects ()

Draws animated objects.

Private Attributes

std::function< void()> m_loopFunction {[]() {}}

custom function fired in the main loop.

RenderWindow m_window {}

Window to draw stuff on.

Point2d m_resoltuon {1000, 800}

Resolution of the window.

• std::string m_windowTitle {"dev"}

Window title.

int m_maxFPS {60}

Max FPS.

• std::array< eventHandler_t, Event::Count > m_eventHandlers

Custom functions handling events.

drawableCollection_t m_drawablesCollection {}

Stuff that is drawn in the window each frame.

- animatedObjecsCollection_t m_animatedObjectsCollection {}
- Clock m_clock {}

Clock for computing ticks.

Time m_lastFrameDuration {}

Duration of the last frame.

Static Private Attributes

static Engine * s_instancePtr {nullptr}

Pointer to the instance.

8.6.1 Detailed Description

The engine class.

Definition at line 18 of file engine.hpp.

8.6.2 Member Typedef Documentation

8.6.2.1 animatedObjecsCollection_t

```
using Engine::animatedObjecsCollection_t = std::set<AnimatedObject *>
```

Definition at line 29 of file engine.hpp.

8.6.2.2 Clock

```
using Engine::Clock = sf::Clock
```

Definition at line 21 of file engine.hpp.

8.6.2.3 Drawable

```
using Engine::Drawable = sf::Drawable
```

Definition at line 24 of file engine.hpp.

8.6.2.4 drawableCollection_t

```
using Engine::drawableCollection_t = std::set<Drawable *>
```

Definition at line 28 of file engine.hpp.

8.6.2.5 Event

```
using Engine::Event = sf::Event
```

Definition at line 23 of file engine.hpp.

8.6.2.6 eventHandler_t

```
using Engine::eventHandler_t = std::function<void(const Event &)>
```

Definition at line 27 of file engine.hpp.

8.6.2.7 RenderWindow

```
using Engine::RenderWindow = sf::RenderWindow
```

Definition at line 22 of file engine.hpp.

8.6.2.8 Shape

```
using Engine::Shape = sf::Shape
```

Definition at line 25 of file engine.hpp.

8.6.2.9 Time

```
using Engine::Time = sf::Time
```

Definition at line 20 of file engine.hpp.

8.6.3 Constructor & Destructor Documentation

8.6.3.1 Engine()

```
Engine::Engine ( )
```

Constructor.

Fills custom event handlers with placeholder functions.

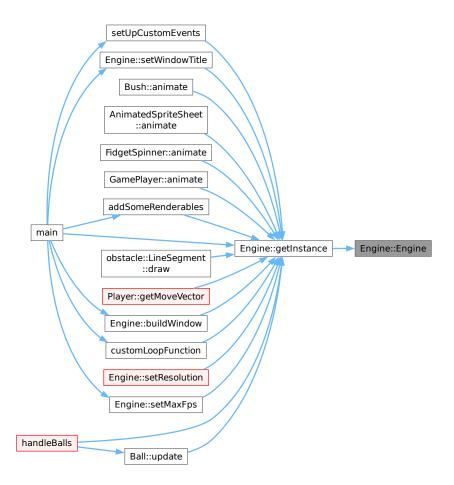
```
Definition at line 14 of file engine.cpp.
```

```
00014 {
00015 LOGINFON;
00017 m_eventHandlers.fill([](const sf::Event &event) {});
00018 }
```

References LOGINFON, and m_eventHandlers.

Referenced by getInstance().

Here is the caller graph for this function:



8.6.3.2 \sim Engine()

```
Engine::\simEngine ( )
```

Destructor.

Definition at line 20 of file engine.cpp.

References LOGINFON, m_animatedObjectsCollection, and m_drawablesCollection.

8.6.4 Member Function Documentation

8.6.4.1 add() [1/2] void Engine::add (

AnimatedObject * animatedObject)

Adds animated object to Collection.

Parameters

animatedObject	animated object.
----------------	------------------

Definition at line 127 of file engine.cpp.

```
00127 {
00128 LOGINFO « animatedObject « '\n';
00129 m_animatedObjectsCollection.insert(animatedObject);
00130 }
```

References LOGINFO, and m_animatedObjectsCollection.

8.6.4.2 add() [2/2]

Adds drawable to Collection.

Parameters

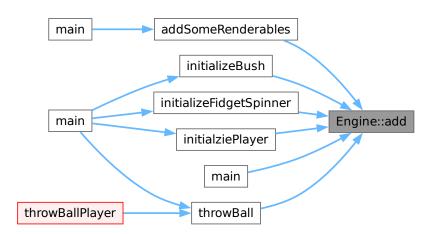
drawable	Drawable shape.
----------	-----------------

Definition at line 122 of file engine.cpp.

```
00122
00123 LOGINFO « drawable « '\n';
00124 m_drawablesCollection.insert(drawable);
00125 }
```

References LOGINFO, and m_drawablesCollection.

Referenced by addSomeRenderables(), initializeBush(), initializeFidgetSpinner(), initialziePlayer(), main(), and throwBall().



8.6.4.3 animateObjects()

```
void Engine::animateObjects ( ) [private]
```

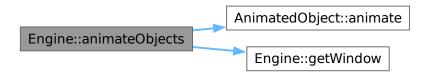
Draws animated objects.

Definition at line 145 of file engine.cpp.

References AnimatedObject::animate(), getWindow(), and m_animatedObjectsCollection.

Referenced by render().

Here is the call graph for this function:



Here is the caller graph for this function:

```
Engine::loop Engine::render Engine::animateObjects
```

8.6.4.4 buildWindow()

```
Engine & Engine::buildWindow ( )
```

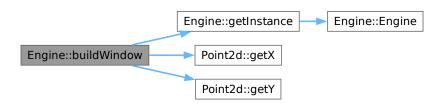
Builds the window.

Definition at line 53 of file engine.cpp.

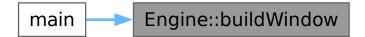
References getInstance(), Point2d::getX(), Point2d::getY(), LOGINFON, m_maxFPS, m_resoltuon, m_window, and m_windowTitle.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.5 clear()

```
void Engine::clear ( )
```

Clears window to a single color.

Definition at line 94 of file engine.cpp. 00094 { m_window.clear(); }

References m_window.

Referenced by loop().



8.6.4.6 display()

```
void Engine::display ( )
```

Displays rendered objects.

Definition at line 101 of file engine.cpp.

```
00101 { m_window.display(); }
```

References m window.

Referenced by loop().

Here is the caller graph for this function:



8.6.4.7 drawDrawables()

```
void Engine::drawDrawables ( ) [private]
```

Draws objects.

Definition at line 140 of file engine.cpp.

References getWindow(), and m_drawablesCollection.

Referenced by render().

Here is the call graph for this function:





8.6.4.8 getInstance()

```
Engine & Engine::getInstance ( ) [static]
```

Instance getter.

Definition at line 30 of file engine.cpp.

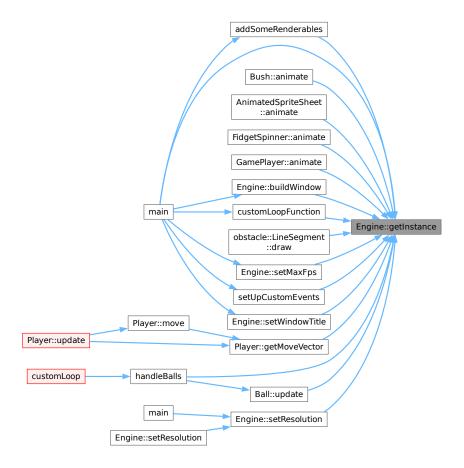
References Engine(), and s_instancePtr.

 $Referenced \ by \ add Some Renderables(), Bush::animate(), Animated Sprite Sheet::animate(), Fidget Spinner::animate(), Game Player::animate(), build Window(), custom Loop Function(), obstacle::Line Segment::draw(), Player::get Move Vector(), handle Balls(), main(), set MaxFps(), set Resolution(), set Up Custom Events(), set Window Title(), and Ball::update().$

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.9 getLastFrameDuration()

```
Engine::Time Engine::getLastFrameDuration ( ) const
```

Last frame duration getter.

```
Definition at line 136 of file engine.cpp.

00136

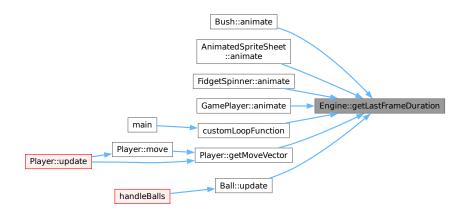
00137 return m_lastFrameDuration;

00138 }
```

References m_lastFrameDuration.

Referenced by Bush::animate(), AnimatedSpriteSheet::animate(), FidgetSpinner::animate(), GamePlayer::animate(), customLoopFunction(), Player::getMoveVector(), and Ball::update().

Here is the caller graph for this function:



8.6.4.10 getResolution()

```
Point2d Engine::getResolution ( ) const
```

Resolution getter.

Definition at line 68 of file engine.cpp. 00068 { return m_resoltuon; }

References m_resoltuon.

8.6.4.11 getWindow()

```
Engine::RenderWindow & Engine::getWindow ( )
```

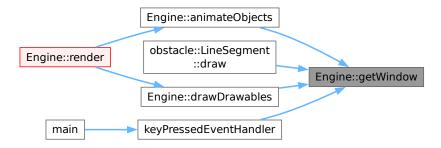
Window getter.

Definition at line 120 of file engine.cpp. 00120 { return m_window; }

References m window.

Referenced by animateObjects(), obstacle::LineSegment::draw(), drawDrawables(), and keyPressedEventHandler().

Here is the caller graph for this function:



8.6.4.12 handleEvents()

```
void Engine::handleEvents ( )
```

Handles events.

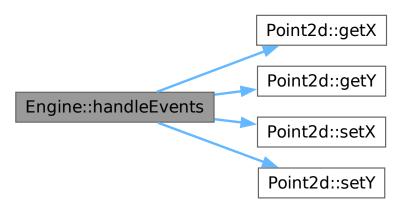
Definition at line 70 of file engine.cpp.

```
sf::Event event;
00071
00072
       while (m_window.pollEvent(event)) {
00073
         if (event.type == sf::Event::Closed)
00074
          m_window.close();
00075
        if (event.type == sf::Event::Resized) {
00076
00077
          // Update engine info
00078
          m_resoltuon.setX(event.size.width);
00079
          m_resoltuon.setY(event.size.height);
08000
           // make new view
          00081
00082
00083
          m_window.setView(sf::View{view});
00084
          LOGINFO « "Resized\t" « event.size.width « '\t' « event.size.height
00085
                  « '\n';
00086
00087
00088
00089
         // Fire custom event handler.
00090
        m_eventHandlers[event.type] (event);
00091
00092 }
```

References Point2d::getX(), Point2d::getY(), LOGINFO, m_eventHandlers, m_resoltuon, m_window, Point2d::setX(), and Point2d::setY().

Referenced by loop().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.13 loop()

```
void Engine::loop ( )
```

The main loop.

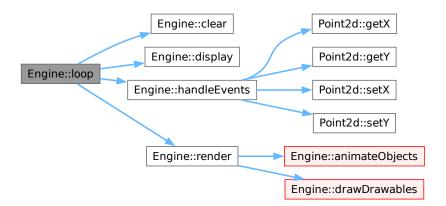
Definition at line 103 of file engine.cpp.

```
00103
00104
        LOGINFON;
00105
00106
        while (m_window.isOpen()) {
00107
00108
         handleEvents();
00109
          // restart clock
00110
00111
          m_lastFrameDuration = m_clock.restart();
00112
00113
          m_loopFunction();
00114
00115
          render();
00116
          display();
00117
```

References clear(), display(), handleEvents(), LOGINFON, m_clock, m_lastFrameDuration, m_loopFunction, m_window, and render().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.14 remove()

References m_drawablesCollection.

Referenced by handleBalls().

Here is the caller graph for this function:



8.6.4.15 render()

```
void Engine::render ( )
```

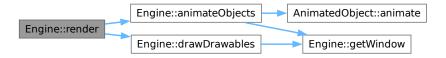
Renders objects.

Definition at line 96 of file engine.cpp.

References animateObjects(), and drawDrawables().

Referenced by loop().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.16 setEventHandler()

Set custom event handler.

Parameters

eventType	Type of event that handler receives.
handler	Function handling the event.

Definition at line 62 of file engine.cpp.

```
00063
00064 m_eventHandlers[eventType] = handler;
```

```
00065    return *this;
00066 }
```

References m_eventHandlers.

Referenced by main(), and setUpCustomEvents().

Here is the caller graph for this function:



8.6.4.17 setLoopFunction()

Set function that is invoked in the main loop.

Parameters

f	function	Function that will be called in loop.
---	----------	---------------------------------------

Definition at line 114 of file engine.hpp.

References m_loopFunction.

Referenced by main().

Here is the caller graph for this function:



8.6.4.18 setMaxFps()

Set Max FPS.

Parameters

```
fps max FPS.
```

Definition at line 36 of file engine.cpp.

```
00036
00037    LOGINFON;
00038    m_maxFPS = fps;
00039    m_window.setFramerateLimit(fps);
00040    return getInstance();
00041 }
```

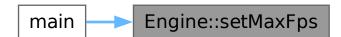
References getInstance(), LOGINFON, m_maxFPS, and m_window.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.19 setResolution() [1/2]

Set resolution.

Parameters

resolution Resolution to be set.

Definition at line 48 of file engine.cpp.

```
00048
00049 m_resoltuon = resolution;
00050 return getInstance();
00051 }
```

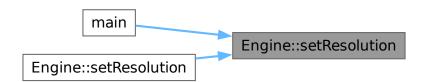
References getInstance(), and m_resoltuon.

Referenced by main(), and setResolution().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.4.20 setResolution() [2/2]

Set resolution.

Definition at line 106 of file engine.hpp.

References setResolution().

Here is the call graph for this function:

```
Engine::setResolution Engine::getInstance
```

8.6.4.21 setWindowTitle()

Set window title.

Parameters



Definition at line 43 of file engine.cpp.

```
00044
        m_windowTitle = title;
00045
        return getInstance();
00046 }
```

References getInstance(), and m_windowTitle.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.5 Member Data Documentation

8.6.5.1 m_animatedObjectsCollection

```
animatedObjecsCollection_t Engine::m_animatedObjectsCollection {} [private]
```

Definition at line 64 of file engine.hpp. 00064 {};

Referenced by add(), animateObjects(), and ~Engine().

8.6.5.2 m_clock

```
Clock Engine::m_clock {} [private]
```

Clock for computing ticks.

Definition at line 68 of file engine.hpp. 00068 {};

Referenced by loop().

8.6.5.3 m_drawablesCollection

```
drawableCollection_t Engine::m_drawablesCollection {} [private]
```

Stuff that is drawn in the window each frame.

```
Definition at line 62 of file engine.hpp. 00062 \ \{\};
```

Referenced by add(), drawDrawables(), remove(), and ~Engine().

8.6.5.4 m eventHandlers

```
std::array<eventHandler_t, Event::Count> Engine::m_eventHandlers [private]
```

Custom functions handling events.

Definition at line 57 of file engine.hpp.

Referenced by Engine(), handleEvents(), and setEventHandler().

8.6.5.5 m_lastFrameDuration

```
Time Engine::m_lastFrameDuration {} [private]
```

Duration of the last frame.

Definition at line 73 of file engine.hpp.

Referenced by getLastFrameDuration(), and loop().

8.6.5.6 m_loopFunction

```
std::function<void() > Engine::m_loopFunction {[]() {}} [private]
```

custom function fired in the main loop.

```
Definition at line 38 of file engine.hpp. 00038 {[]() {}};
```

Referenced by loop(), and setLoopFunction().

8.6.5.7 m maxFPS

```
int Engine::m_maxFPS {60} [private]
```

Max FPS.

Definition at line 52 of file engine.hpp. 00052 {60};

Referenced by buildWindow(), and setMaxFps().

8.6.5.8 m_resoltuon

```
Point2d Engine::m_resoltuon {1000, 800} [private]
```

Resolution of the window.

Definition at line 45 of file engine.hpp. 00045 {1000, 800};

Referenced by buildWindow(), getResolution(), handleEvents(), and setResolution().

8.6.5.9 m_window

```
RenderWindow Engine::m_window {} [private]
```

Window to draw stuff on.

Definition at line 42 of file engine.hpp.

Referenced by buildWindow(), clear(), display(), getWindow(), handleEvents(), loop(), and setMaxFps().

8.6.5.10 m_windowTitle

```
std::string Engine::m_windowTitle {"dev"} [private]
```

Window title.

Definition at line 48 of file engine.hpp. 00048 {"dev"};

Referenced by buildWindow(), and setWindowTitle().

8.6.5.11 s_instancePtr

```
Engine * Engine::s_instancePtr {nullptr} [static], [private]
```

Pointer to the instance.

Definition at line 12 of file engine.hpp.

Referenced by getInstance().

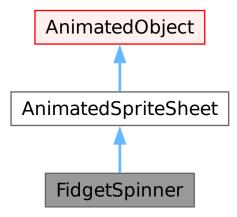
The documentation for this class was generated from the following files:

- · engine.hpp
- · engine.cpp

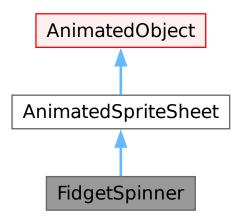
8.7 FidgetSpinner Class Reference

#include <fidgetSpinner.hpp>

Inheritance diagram for FidgetSpinner:



Collaboration diagram for FidgetSpinner:



Public Types

using animationData_t = std::vector< AnimationFrameData >

Public Member Functions

- FidgetSpinner (std::string_view path)
- virtual void animate () override

Animates object.

- · Point2d getCenterPoint () const
- void addRotationSpeed (float speed)
- void setRotationResistance (float speed)
- virtual void setPosition (Point2d pos)

Object position setter.

virtual void move (Point2d vec)

Translates object in space.

· virtual Point2d getPosition () const

Object position getter.

Protected Member Functions

- · int getCurrentAnimationFrameCount () const
- sf::IntRect getCurrentAnimationFrameRect () const
- float getCurrentAnimationFrameDuration () const
- const AnimationFrameData & getCurrentAnimationFrameData () const
- void nextFrame ()
- void setFrame (const AnimationFrameData &frameData)
- void loadFrame (std::istream &stream, animationData_t &animationData)

load frame data from stream

- void loadSpritesheet (std::istream &stream, std::string_view configFileDirectoryPath)
- void loadFromConfigFile (std::string_view pathToDir)

Private Attributes

- float m_rotationResistance {0.1}
- float m_rotationspeed {0}
- float m_angle {}
- std::vector< animationData_t > m_animationsData
- int m_currentAnimationTypeIndex
- int m_currentFrameId {}
- float m_animationTimer {}
- Point2d m_position {}

Position of object on screen.

8.7.1 Detailed Description

Definition at line 8 of file fidgetSpinner.hpp.

8.7.2 Member Typedef Documentation

8.7.2.1 animationData_t

using AnimatedSpriteSheet::animationData_t = std::vector<AnimationFrameData> [inherited]

Definition at line 24 of file animatedSpriteSheet.hpp.

8.7.3 Constructor & Destructor Documentation

8.7.3.1 FidgetSpinner()

References LOGINFON, and Point2d::toVector2f().

Here is the call graph for this function:



8.7.4 Member Function Documentation

8.7.4.1 addRotationSpeed()

References m_rotationspeed.

00035 { m_rotationspeed += speed; }

Referenced by spinTheFidget().

Here is the caller graph for this function:



8.7.4.2 animate()

```
void FidgetSpinner::animate ( ) [override], [virtual]
```

Animates object.

Reimplemented from AnimatedSpriteSheet.

Definition at line 16 of file fidgetSpinner.cpp.

```
00017
        m\_angle += m\_rotationspeed *
                   Engine::getInstance().getLastFrameDuration().asSeconds();
00018
00019
       setRotation(m_angle);
00020
00021
       if (m_rotationspeed > 0) {
00022
         m_rotationspeed -= m_rotationResistance + m_rotationspeed / 500;
        if (m_rotationspeed < 0)</pre>
00023
00024
            m_rotationspeed = 0;
00025
       }
00026 }
```

References Engine::getInstance(), Engine::getLastFrameDuration(), m_angle, m_rotationResistance, and m_rotationspeed.

Here is the call graph for this function:



8.7.4.3 getCenterPoint()

```
Point2d FidgetSpinner::getCenterPoint ( ) const
```

Definition at line 28 of file fidgetSpinner.cpp.

```
00028 {
00029 Point2d point{getCurrentAnimationFrameData().m_position +
00030 getCurrentAnimationFrameData().m_size * 0.5};
00031
00032 return point;
00033 }
```

 $References\ AnimatedSpriteSheet:: AnimationFrameData::m_position, and\ AnimatedSpriteSheet:: AnimationFrameData::m_size.$

Here is the call graph for this function:



8.7.4.4 getCurrentAnimationFrameCount()

int AnimatedSpriteSheet::getCurrentAnimationFrameCount () const [protected], [inherited]

Definition at line 54 of file animatedSpriteSheet.cpp.

```
00054
00055 LOGTRACEN;
00056 return m_animationsData[m_currentAnimationTypeIndex].size();
00057 }
```

References LOGTRACEN, AnimatedSpriteSheet::m_animationsData, and AnimatedSpriteSheet::m_currentAnimationTypeIndex.

Referenced by AnimatedSpriteSheet::nextFrame().

Here is the caller graph for this function:



8.7.4.5 getCurrentAnimationFrameData()

```
const AnimatedSpriteSheet::AnimationFrameData & AnimatedSpriteSheet::getCurrentAnimation←
FrameData ( ) const [protected], [inherited]
```

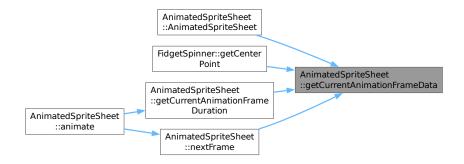
Definition at line 60 of file animatedSpriteSheet.cpp.

```
00060
00061
        LOGTRACEN;
00062
        if (m_animationsData.size() == 0 ||
00063
            m_animationsData.size() < m_currentAnimationTypeIndex)</pre>
00064
          LOGWARN « "No animation data\n";
00065
        } else if (m_animationsData[m_currentAnimationTypeIndex].size() == 0 ||
00066
                   m_animationsData[m_currentAnimationTypeIndex].size() <</pre>
00067
                       m currentAnimationTvpeIndex) {
00068
          LOGWARN « "No frame data in animation " « m_currentAnimationTypeIndex
                  « '\n';
00070
00071
        return m_animationsData[m_currentAnimationTypeIndex][m_currentFrameId];
00072 }
```

References LOGTRACEN, LOGWARN, AnimatedSpriteSheet::m_animationsData, AnimatedSpriteSheet::m_currentAnimationTypeIn and AnimatedSpriteSheet::m_currentFrameId.

Referenced by AnimatedSpriteSheet::AnimatedSpriteSheet(), getCenterPoint(), AnimatedSpriteSheet::getCurrentAnimationFrameDu and AnimatedSpriteSheet::nextFrame().

Here is the caller graph for this function:



8.7.4.6 getCurrentAnimationFrameDuration()

float AnimatedSpriteSheet::getCurrentAnimationFrameDuration () const [protected], [inherited]

```
Definition at line 74 of file animatedSpriteSheet.cpp.

00074

00075 LOGTRACEN;

00076 return getCurrentAnimationFrameData().m_duration;
```

 $References\ Animated Sprite Sheet:: get Current Animation Frame Data(), LOGTRACEN, and Animated Sprite Sheet:: Animation Frame Data(), LOGTRACEN, and Animation Frame Data(), LOGTRACEN, Ani$

Referenced by AnimatedSpriteSheet::animate().

Here is the call graph for this function:

00077 };



Here is the caller graph for this function:



8.7.4.7 getCurrentAnimationFrameRect()

```
sf::IntRect AnimatedSpriteSheet::getCurrentAnimationFrameRect ( ) const [protected], [inherited]
```

8.7.4.8 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

```
Definition at line 7 of file GameObject.cpp.
00007 { return m_position; }
```

References GameObject::m position.

8.7.4.9 loadFrame()

load frame data from stream

Definition at line 85 of file animatedSpriteSheet.cpp.

```
00087
00088
        // load Frame data
00089
        AnimationFrameData fdat{};
00090
        stream » fdat;
        // LOGINFO « fdat.m_position « " " « fdat.m_size « " " «
00091
00092
        // fdat.m_duration
00093
                   « '\n';
00094
00095
        // add frame data to animation
00096
        animationData.push_back(fdat);
        \verb|stream.ignore(std::numeric_limits < std::streamsize > :: max(), \ ' \ ');|
00097
00098 };
```

References LOGTRACEN.

Referenced by AnimatedSpriteSheet::loadFromConfigFile().

Here is the caller graph for this function:



8.7.4.10 loadFromConfigFile()

Definition at line 113 of file animatedSpriteSheet.cpp.

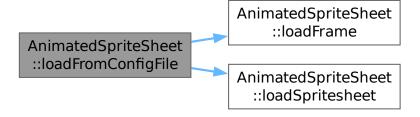
```
00114
        std::string configFilePath{static_cast<std::string>(pathToDir) +
00115
                                       "/config.txt"};
        // open file
00116
        std::fstream configFile{configFilePath};
00117
00118
        if (!configFile.is_open()) {
00119
          LOGERROR « "opening config file failed it should be at " « configFilePath
00120
00121
          return;
00122
00123
00124
        animationData_t *animationDataBeingLoaded{NULL};
00125
        std::string line{};
00126
00127
        while (!configFile.eof()) {
  line = "";
00128
00129
           \ensuremath{//} load line, ignore lines that are commnets
00130
          do {
00131
            std::getline(configFile, line);
00132
          } while (line.starts_with("#"));
00133
00134
           // load spritesheet
          if (line.starts_with("SPRITESHEET")) {
00135
           loadSpritesheet(configFile, pathToDir);
} else if (line.starts_with("ANIMATION")) {
00136
00137
00138
             // Add new animation and change pointer
```

```
00139
                     LOGINFO « "loading animation data\n";
00140
             m_animationsData.push_back({});
00141
             animationDataBeingLoaded = {&m_animationsData.back()};
00142
           // load frame
00143
00144
           else if (line.starts_with("FRAME")) {
00145
             if (animationDataBeingLoaded == NULL)
00146
               LOGERROR « "config file is corrupted, FRAME before ANIMATION?";
00147
             loadFrame(configFile, *animationDataBeingLoaded);
00148
           // not known
00149
00150
          else {
  // empty line ignore
  if (line == "") {
00151
00152
00153
00154
             // invalid line
00155
             LOGWARN « "not recegnized config option: " « line « " in file: " « configFilePath « '\n';
00156
00157
00158
00159
        // close file
00160
00161
        configFile.close();
00162 }
```

References AnimatedSpriteSheet::loadFrame(), AnimatedSpriteSheet::loadSpritesheet(), LOGERROR, LOGWARN, and AnimatedSpriteSheet::m_animationsData.

Referenced by AnimatedSpriteSheet::AnimatedSpriteSheet().

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.4.11 loadSpritesheet()

Definition at line 100 of file animatedSpriteSheet.cpp.

```
LOGINFO « "loading spritesheet data\n";
00102
00103
        std::string spriteSheetRelPath{};
        stream » spriteSheetRelPath;
00104
00105
        spriteSheetRelPath =
00106
            std::string(configFileDirectoryPath) + "/" + spriteSheetRelPath;
00107
00108
        loadFromFile(spriteSheetRelPath);
00109
       stream.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00110
00111 }
```

References LOGINFO.

Referenced by AnimatedSpriteSheet::loadFromConfigFile().

Here is the caller graph for this function:



8.7.4.12 move()

Translates object in space.

Parameters

vector	2D vector added to current object position.
--------	---

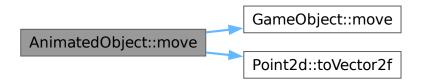
Reimplemented from GameObject.

Reimplemented in Player.

Definition at line 14 of file animatedObject.cpp.

References GameObject::move(), and Point2d::toVector2f().

Here is the call graph for this function:



8.7.4.13 nextFrame()

```
void AnimatedSpriteSheet::nextFrame ( ) [protected], [inherited]
```

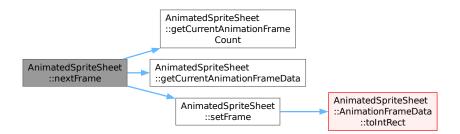
Definition at line 46 of file animatedSpriteSheet.cpp.

```
00046
00047 LOGTRACEN;
00048  // change frame
00049  ++m_currentFrameId;
00050  m_currentFrameId %= getCurrentAnimationFrameCount();
00051  setFrame(getCurrentAnimationFrameData());
00052 }
```

References AnimatedSpriteSheet::getCurrentAnimationFrameCount(), AnimatedSpriteSheet::getCurrentAnimationFrameData(), LOGTRACEN, AnimatedSpriteSheet::m currentFrameId, and AnimatedSpriteSheet::setFrame().

Referenced by AnimatedSpriteSheet::animate().

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.4.14 setFrame()

References LOGTRACEN, AnimatedSpriteSheet::m_animationTimer, and AnimatedSpriteSheet::AnimationFrameData::toIntRect().

Referenced by AnimatedSpriteSheet::AnimatedSpriteSheet(), and AnimatedSpriteSheet::nextFrame().

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.4.15 setPosition()

Object position setter.

Parameters

pos	Position to be set.

Reimplemented from GameObject.

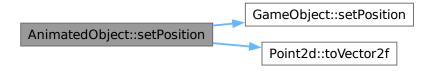
Definition at line 9 of file animatedObject.cpp.

```
00009
00010    GameObject::setPosition(pos);
00011    sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.4.16 setRotationResistance()

References m_rotationResistance.

8.7.5 Member Data Documentation

8.7.5.1 m_angle

```
float FidgetSpinner::m_angle {} [private]

Definition at line 24 of file fidgetSpinner.hpp.
00024 {};
```

Referenced by animate().

8.7.5.2 m_animationsData

```
std::vector<animationData_t> AnimatedSpriteSheet::m_animationsData [private], [inherited]
```

Colecction of animations.

Definition at line 55 of file animatedSpriteSheet.hpp.

Referenced by AnimatedSpriteSheet::getCurrentAnimationFrameCount(), AnimatedSpriteSheet::getCurrentAnimationFrameData(), and AnimatedSpriteSheet::loadFromConfigFile().

8.7.5.3 m animationTimer

```
float AnimatedSpriteSheet::m_animationTimer {} [private], [inherited]
```

Definition at line 63 of file animatedSpriteSheet.hpp.

Referenced by AnimatedSpriteSheet::animate(), and AnimatedSpriteSheet::setFrame().

8.7.5.4 m_currentAnimationTypeIndex

```
int AnimatedSpriteSheet::m_currentAnimationTypeIndex [private], [inherited]
```

Indicates index of current animation in m animationsData.

Definition at line 60 of file animatedSpriteSheet.hpp.

Referenced by AnimatedSpriteSheet::getCurrentAnimationFrameCount(), and AnimatedSpriteSheet::getCurrentAnimationFrameData

8.7.5.5 m_currentFrameId

```
int AnimatedSpriteSheet::m_currentFrameId {} [private], [inherited]
```

Definition at line 61 of file animatedSpriteSheet.hpp.

Referenced by AnimatedSpriteSheet::getCurrentAnimationFrameData(), and AnimatedSpriteSheet::nextFrame().

8.7.5.6 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

Definition at line 14 of file GameObject.hpp. 00014 {};

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

8.7.5.7 m_rotationResistance

```
float FidgetSpinner::m_rotationResistance {0.1} [private]
```

Definition at line 22 of file fidgetSpinner.hpp. 00022 {0.1};

Referenced by animate(), and setRotationResistance().

8.7.5.8 m_rotationspeed

```
float FidgetSpinner::m_rotationspeed {0} [private]
```

Definition at line 23 of file fidgetSpinner.hpp. 00023 {0};

Referenced by addRotationSpeed(), and animate().

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

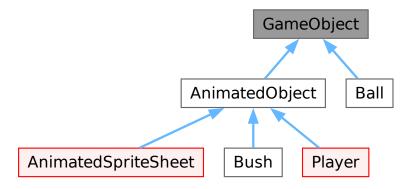
- fidgetSpinner.hpp
- · fidgetSpinner.cpp

8.8 GameObject Class Reference

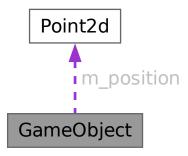
Game object class.

```
#include <GameObject.hpp>
```

Inheritance diagram for GameObject:



Collaboration diagram for GameObject:



Public Member Functions

virtual ∼GameObject ()

Destructor.

virtual void setPosition (Point2d pos)

Object position setter.

• virtual Point2d getPosition () const

Object position getter.

virtual void move (Point2d vector)

Translates object in space.

Private Attributes

Point2d m_position {}

Position of object on screen.

8.8.1 Detailed Description

Game object class.

Definition at line 9 of file GameObject.hpp.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 ~GameObject()

```
virtual GameObject::~GameObject ( ) [inline], [virtual]
```

Destructor.

Definition at line 20 of file GameObject.hpp. 00020 {};

8.8.3 Member Function Documentation

8.8.3.1 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual]
```

Object position getter.

Definition at line 7 of file GameObject.cpp.

```
00007 { return m_position; }
```

References m_position.

8.8.3.2 move()

Translates object in space.

Parameters

```
vector 2D vector added to current object position.
```

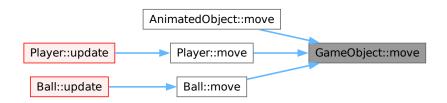
Reimplemented in AnimatedObject, Player, and Ball.

Definition at line 9 of file GameObject.cpp.

References m_position.

Referenced by AnimatedObject::move(), Player::move(), and Ball::move().

Here is the caller graph for this function:



8.8.3.3 setPosition()

Object position setter.

Parameters

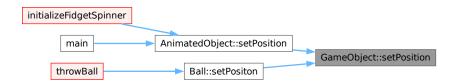
Reimplemented in AnimatedObject.

Definition at line 5 of file GameObject.cpp. 00005 { m_position = pos; }

References m_position.

Referenced by AnimatedObject::setPosition(), and Ball::setPositon().

Here is the caller graph for this function:



8.8.4 Member Data Documentation

8.8.4.1 m_position

Point2d GameObject::m_position {} [private]

Position of object on screen.

Definition at line 14 of file GameObject.hpp.

Referenced by getPosition(), move(), and setPosition().

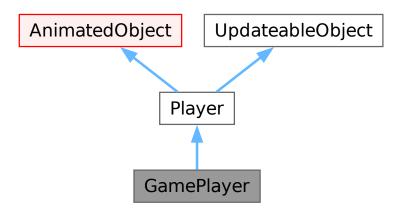
The documentation for this class was generated from the following files:

- GameObject.hpp
- · GameObject.cpp

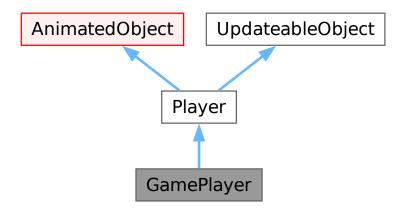
8.9 GamePlayer Class Reference

#include <gamePlayer.hpp>

Inheritance diagram for GamePlayer:



Collaboration diagram for GamePlayer:



Public Types

```
    enum class AnimationType {
    standing , moveNorth , moveEast , moveSouth ,
    moveWest , moveSpecial , count }
    enum class MoveDirection {
```

north, east, south, west, count}

Directions of movement.

Public Member Functions

- · GamePlayer ()
- virtual void animate () override

Animates object.

· virtual void update () override

Updates state of object.

- virtual void setIsMoving (MoveDirection direction)
- virtual void stopMoving (MoveDirection direction)
- void stopMoving ()

Stops player from moving.

• bool isMoving () const

Checks if player is moving.

• bool isMoving (MoveDirection direction) const

Checks if player is moving in a specific direction.

void setMovementSpeed (float speed)

Player movement speed setter.

virtual float getMovementSpeed () const

Player movement speed getter.

· bool isMovingDiagonaly () const

Checks if player is moving diagonally.

• Point2d getMoveVectorOrigin () const

Returns a 2D vector containing information on which cardinal directions the player is moving.

Point2d getMoveVector () const

Returns the player's movement vector multiplied by speed.

virtual void setPosition (Point2d pos)

Object position setter.

· virtual Point2d getPosition () const

Object position getter.

Static Public Attributes

• static constexpr std::string_view s_spriteSheetPath

Private Member Functions

- · void nextAnimationFrame ()
- void updateTextureRect ()
- void setAnimationType (AnimationType type)
- AnimationType getAnimationTypeOf (MoveDirection moveDirection) const
- int getFrameCountOfAnimation (AnimationType type) const
- sf::IntRect getCurrentAnimationFrameSpriteRectangle () const
- int getRowOfAnimation () const
- void move (Point2d vec)

Moves player on screen.

Private Attributes

- int m_aninmationFrameIndicator {}
- float m_animationFrameDuration {.5}
- float m_animationTimer {}
- AnimationType m_animationType {AnimationType::standing}
- float m_movementSpeed {50}

Player movement speed.

• std::array< bool, static_cast< ssize_t >(MoveDirection::count)> m_isMoving {}

Array with player's current movement direction.

Point2d m_position {}

Position of object on screen.

8.9.1 Detailed Description

Definition at line 10 of file gamePlayer.hpp.

8.9.2 Member Enumeration Documentation

8.9.2.1 AnimationType

```
enum class GamePlayer::AnimationType [strong]
```

Enumerator

standing	
moveNorth	
moveEast	
moveSouth	
moveWest	
moveSpecial	
count	

Definition at line 15 of file gamePlayer.hpp.

```
00015

00016 standing,

00017 moveNorth,

00018 moveEast,

00019 moveSouth,

00020 moveWest,

00021 moveSpecial,

00022 count

00023 };
```

8.9.2.2 MoveDirection

```
enum class Player::MoveDirection [strong], [inherited]
```

Directions of movement.

Enumerator

north	
east	
south	
west	
count	

Definition at line 21 of file player.hpp.

```
00021 { north, east, south, west, count };
```

8.9.3 Constructor & Destructor Documentation

8.9.3.1 GamePlayer()

```
GamePlayer::GamePlayer ( )
```

Definition at line 15 of file gamePlayer.cpp.

References count, getCurrentAnimationFrameSpriteRectangle(), LOGINFON, and s_spriteSheetPath.

Here is the call graph for this function:



8.9.4 Member Function Documentation

8.9.4.1 animate()

```
void GamePlayer::animate ( ) [override], [virtual]
```

Animates object.

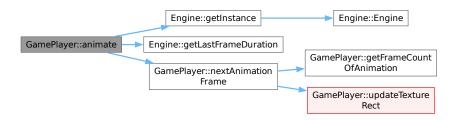
Reimplemented from AnimatedObject.

Definition at line 31 of file gamePlayer.cpp.

```
00031
00032
         m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
        // wait for next frame change
if (m_animationTimer < m_animationFrameDuration)
00033
00034
00035
            return;
         // reset timer
00036
        m_animationTimer = 0;
// do frame changing
00037
00038
00039
        nextAnimationFrame();
00040 }
```

References Engine::getInstance(), Engine::getLastFrameDuration(), m_animationFrameDuration, m_animationTimer, and nextAnimationFrame().

Here is the call graph for this function:



8.9.4.2 getAnimationTypeOf()

```
GamePlayer::AnimationType GamePlayer::getAnimationTypeOf (
                MoveDirection moveDirection ) const [private]
Definition at line 99 of file gamePlayer.cpp.
00099
00100
         switch (moveDirection) {
        case Player::MoveDirection::north:
    return AnimationType::moveNorth;
00101
00102
00103
         case Player::MoveDirection::east:
00104
          return AnimationType::moveEast;
00105
        case Player::MoveDirection::south:
         return AnimationType::moveSouth;
case Player::MoveDirection::west:
00106
00107
00108
          return AnimationType::moveWest;
00109
00110
           LOGWARN « "UNHANDLED\n";
00111
          break;
00112
00113
         return AnimationType::standing;
00114 }
```

References Player::east, LOGWARN, moveEast, moveNorth, moveSouth, moveWest, Player::north, Player::south, standing, and Player::west.

Referenced by update().

Here is the caller graph for this function:



8.9.4.3 getCurrentAnimationFrameSpriteRectangle()

sf::IntRect GamePlayer::getCurrentAnimationFrameSpriteRectangle () const [private]

Definition at line 24 of file gamePlayer.cpp.

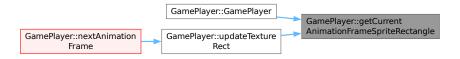
References getRowOfAnimation(), and m aninmationFrameIndicator.

Referenced by GamePlayer(), and updateTextureRect().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.4.4 getFrameCountOfAnimation()

Definition at line 67 of file gamePlayer.cpp.

```
00067
00068
        switch (type) {
00069
        case AnimationType::standing:
00070
         return 4;
00071
        case AnimationType::moveNorth:
00072
00073
        case AnimationType::moveEast:
00074
         return 4:
00075
        case AnimationType::moveSouth:
00076
         return 4;
00077
        case AnimationType::moveWest:
00078
         return 4;
00079
        case AnimationType::moveSpecial:
08000
        return 4;
default:
00081
00082
         LOGWARN « "Animation type not handled!\n";
00083
          return 0;
00084
00085
       }
00086 }
```

References LOGWARN, moveEast, moveNorth, moveSouth, moveSpecial, moveWest, and standing.

Referenced by nextAnimationFrame().

Here is the caller graph for this function:



8.9.4.5 getMovementSpeed()

```
float Player::getMovementSpeed ( ) const [virtual], [inherited]
```

Player movement speed getter.

```
Definition at line 32 of file player.cpp.
00032 { return m_movementSpeed; };
```

References Player::m_movementSpeed.

8.9.4.6 getMoveVector()

```
Point2d Player::getMoveVector ( ) const [inherited]
```

Returns the player's movement vector multiplied by speed.

```
Definition at line 40 of file player.cpp.
```

References Engine::getInstance(), Engine::getLastFrameDuration(), Player::getMoveVectorOrigin(), and Player::m_movementSpeed.

Referenced by Player::move(), and Player::update().

Here is the call graph for this function:





8.9.4.7 getMoveVectorOrigin()

```
Point2d Player::getMoveVectorOrigin ( ) const [inherited]
```

Returns a 2D vector containing information on which cardinal directions the player is moving.

Definition at line 17 of file player.cpp.

```
00018
        Point2d vec{};
00019
00020
        if (m_isMoving[static_cast<int>(MoveDirection::north)])
00021
         vec += \{0, -1\};
00022
        if (m_isMoving[static_cast<int>(MoveDirection::east)])
00023
         vec += \{1, 0\};
00024
        if (m_isMoving[static_cast<int>(MoveDirection::south)])
00025
         vec += {0, 1};
00026
       if (m_isMoving[static_cast<int>(MoveDirection::west)])
00027
         vec += \{-1, 0\};
00028
       return vec;
00029 }
```

References Player::east, Player::m isMoving, Player::north, Player::south, and Player::west.

Referenced by Player::getMoveVector(), Player::isMoving(), Player::isMovingDiagonaly(), and throwBallPlayer().

Here is the caller graph for this function:



8.9.4.8 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

```
Definition at line 7 of file GameObject.cpp. 00007 { return m_position; }
```

References GameObject::m_position.

8.9.4.9 getRowOfAnimation()

```
int GamePlayer::getRowOfAnimation ( ) const [private]
```

Definition at line 128 of file gamePlayer.cpp.

```
00128
        switch (m_animationType) {
00130
       case AnimationType::standing:
00131
         return 0;
00132
       case AnimationType::moveNorth:
00133
         return 1;
00134
       case AnimationType::moveEast:
00135
         return 2;
00136
       case AnimationType::moveSouth:
```

```
00137
         return 3;
00138
       case AnimationType::moveWest:
00139
         return 4;
00140
       case AnimationType::moveSpecial:
00141
         return 5;
00142
       default:
       LOGWARN « "not handled\n";
00143
00144
         return 0;
00145
       };
00146 };
```

References LOGWARN, m_animationType, moveEast, moveNorth, moveSouth, moveSpecial, moveWest, and standing.

Referenced by getCurrentAnimationFrameSpriteRectangle().

Here is the caller graph for this function:



8.9.4.10 isMoving() [1/2]

```
bool Player::isMoving ( ) const [inherited]
```

Checks if player is moving.

```
Definition at line 53 of file player.cpp.
00053
00054 bool moving{};
00055 return getMoveVectorOrigin() != Point2d{0, 0};
00056 }
```

References Player::getMoveVectorOrigin().

Referenced by update().

Here is the call graph for this function:





8.9.4.11 isMoving() [2/2]

Checks if player is moving in a specific direction.

Parameters

```
direction Direction checked.
```

Definition at line 58 of file player.cpp.

```
00058
00059    return m_isMoving[static_cast<ssize_t>(direction)];
00060 }
```

References Player::m_isMoving.

8.9.4.12 isMovingDiagonaly()

```
bool Player::isMovingDiagonaly ( ) const [inherited]
```

Checks if player is moving diagonally.

Definition at line 34 of file player.cpp.

References Player::getMoveVectorOrigin().

Referenced by update().

Here is the call graph for this function:

```
Player::isMovingDiagonaly Player::getMoveVectorOrigin
```

Here is the caller graph for this function:

```
handlePlayerMovement — GamePlayer::update — Player::isMovingDiagonaly
```

8.9.4.13 move()

Moves player on screen.

Parameters

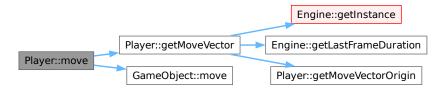
```
vec Vector by which player is moved.
```

Reimplemented from AnimatedObject.

References Player::getMoveVector(), and GameObject::move().

Referenced by Player::update().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.4.14 nextAnimationFrame()

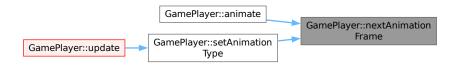
References getFrameCountOfAnimation(), m_animationType, m_animationFrameIndicator, and updateTextureRect().

Referenced by animate(), and setAnimationType().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.4.15 setAnimationType()

```
void GamePlayer::setAnimationType (
                GamePlayer::AnimationType type ) [private]
Definition at line 88 of file gamePlayer.cpp.
00088
00089
        // Do no reset current animation if new is same.
if (type == m_animationType)
00090
00091
          return;
00092
        m_animationType = type;
00093
        m_aninmationFrameIndicator = 0;
00094
        m_animationTimer = 0;
00095
        nextAnimationFrame();
00096 l:
```

 $References\ m_animation Timer,\ m_animation Type,\ m_aninmation Frame Indicator,\ and\ next Animation Frame ().$

Referenced by update().

Here is the call graph for this function:





8.9.4.16 setIsMoving()

Parameters

```
direction
```

References Player::m_isMoving.

Referenced by keyPressedEventHandler().

Here is the caller graph for this function:



8.9.4.17 setMovementSpeed()

Player movement speed setter.

Definition at line 31 of file player.cpp. 00031 { m_movementSpeed = speed; }

References Player::m movementSpeed.

Referenced by initialziePlayer().

Here is the caller graph for this function:



8.9.4.18 setPosition()

Object position setter.

Parameters

```
pos Position to be set.
```

Reimplemented from GameObject.

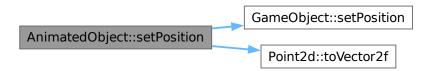
Definition at line 9 of file animatedObject.cpp.

```
00009
00010    GameObject::setPosition(pos);
00011    sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:

```
main initializeFidgetSpinner

AnimatedObject::setPosition
```

8.9.4.19 stopMoving() [1/2]

```
void Player::stopMoving ( ) [inherited]
```

Stops player from moving.

Definition at line 62 of file player.cpp.

References Player::m isMoving.

8.9.4.20 stopMoving() [2/2]

Parameters

direction

Definition at line 49 of file player.cpp.

```
00049
00050 m_isMoving[static_cast<ssize_t>(direction)] = false;
00051 }
```

References Player::m_isMoving.

Referenced by keyReleasedEventHandler().

Here is the caller graph for this function:



8.9.4.21 update()

```
void GamePlayer::update ( ) [override], [virtual]
```

Updates state of object.

Reimplemented from Player.

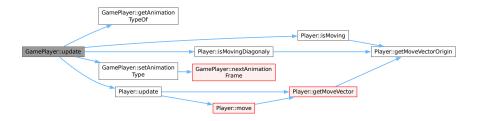
Definition at line 42 of file gamePlayer.cpp.

```
00042
00043
        Player::update();
00044
       // Handle animation update
AnimationType animation = AnimationType::standing;
00045
00046
00047
00048
        if (isMoving()) {
         // if is moving diagonally
if (isMovingDiagonaly()) {
00049
00050
00051
            animation = AnimationType::moveSpecial;
00052
          } else {
00053
            // check if is moving in any single direction
00054
             for (int i{}; i < static_cast<int>(MoveDirection::count); ++i) {
00055
              MoveDirection moveDir{static_cast<MoveDirection>(i)};
00056
               if (isMoving(moveDir)) {
                 animation = getAnimationTypeOf(moveDir);
00057
00058
                 break;
00059
               }
00060
00061
00062
        }
00063
00064
        setAnimationType(animation);
00065 }
```

References animation, Player::count, getAnimationTypeOf(), Player::isMoving(), Player::isMovingDiagonaly(), moveSpecial, setAnimationType(), standing, and Player::update().

Referenced by handlePlayerMovement().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.4.22 updateTextureRect()

```
void GamePlayer::updateTextureRect ( ) [private]
```

Definition at line 124 of file gamePlayer.cpp.

```
00124 {
00125 this->setTextureRect(getCurrentAnimationFrameSpriteRectangle());
00126 }
```

References getCurrentAnimationFrameSpriteRectangle().

Referenced by nextAnimationFrame().

Here is the call graph for this function:





8.9.5 Member Data Documentation

8.9.5.1 m_animationFrameDuration

```
float GamePlayer::m_animationFrameDuration {.5} [private]
```

Definition at line 33 of file gamePlayer.hpp.

Referenced by animate().

8.9.5.2 m_animationTimer

```
float GamePlayer::m_animationTimer {} [private]
```

Definition at line 34 of file gamePlayer.hpp. $00034 \{\};$

Referenced by animate(), and setAnimationType().

8.9.5.3 m_animationType

```
AnimationType GamePlayer::m_animationType {AnimationType::standing} [private]
```

Definition at line 35 of file gamePlayer.hpp. 00035 {AnimationType::standing};

Referenced by getRowOfAnimation(), nextAnimationFrame(), and setAnimationType().

8.9.5.4 m_aninmationFrameIndicator

```
int GamePlayer::m_aninmationFrameIndicator {} [private]
```

Definition at line 32 of file gamePlayer.hpp. 00032 {};

Referenced by getCurrentAnimationFrameSpriteRectangle(), nextAnimationFrame(), and setAnimationType().

8.9.5.5 m_isMoving

```
std::array<bool, static_cast<ssize_t>(MoveDirection::count)> Player::m_isMoving {} [private],
[inherited]
```

Array with player's current movement direction.

Definition at line 31 of file player.hpp.

Referenced by Player::getMoveVectorOrigin(), Player::isMoving(), Player::setIsMoving(), Player::stopMoving(), and Player::stopMoving().

8.9.5.6 m_movementSpeed

```
float Player::m_movementSpeed {50} [private], [inherited]
```

Player movement speed.

```
Definition at line 27 of file player.hpp. 00027 {50};
```

Referenced by Player::getMovementSpeed(), Player::getMoveVector(), and Player::setMovementSpeed().

8.9.5.7 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

```
Definition at line 14 of file GameObject.hpp. 00014 {};
```

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

8.9.5.8 s_spriteSheetPath

```
constexpr std::string_view GamePlayer::s_spriteSheetPath [static], [constexpr]

Initial value:
{
    "resource/player/spritesheet.png"}

Definition at line 12 of file gamePlayer.hpp.
00012
00013    "resource/player/spritesheet.png"};
```

Referenced by GamePlayer().

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

- gamePlayer.hpp
- gamePlayer.cpp

8.10 obstacle::LineSegment Class Reference

```
#include <lineSegment.hpp>
```

Public Member Functions

- LineSegment (Point2d start, Point2d end, sf::Color color={})
- · void draw () const
- · Point2d getStart () const
- Point2d getEnd () const
- void setStart (Point2d val)
- void setEnd (Point2d val)
- void setColor (sf::Color color)

Private Attributes

sf::Vertex m_points [2] = {}sf::Color m_color

8.10.1 Detailed Description

Definition at line 14 of file lineSegment.hpp.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 LineSegment()

References Point2d::toVector2f().

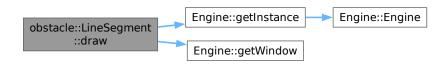
Here is the call graph for this function:



8.10.3 Member Function Documentation

8.10.3.1 draw()

References Engine::getInstance(), and Engine::getWindow().



8.10.3.2 getEnd()

References m_color.

00029 { m_color = color; };

8.10.3.5 setEnd()

```
void obstacle::LineSegment::setEnd ( {\tt Point2d}\ val\ )
```

Definition at line 29 of file lineSegment.hpp.

Definition at line 29 of file lineSegment.cpp.

References Point2d::toVector2f().



8.10.3.6 setStart()

References Point2d::toVector2f().

Here is the call graph for this function:



8.10.4 Member Data Documentation

8.10.4.1 m_color

```
sf::Color obstacle::LineSegment::m_color [private]
```

Definition at line 16 of file lineSegment.hpp.

Referenced by setColor().

8.10.4.2 m_points

```
sf::Vertex obstacle::LineSegment::m_points[2] = {} [private]
```

Definition at line 15 of file lineSegment.hpp. 00015 {};

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

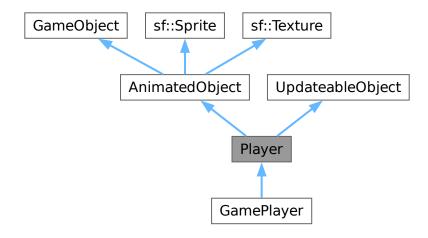
- lineSegment.hpp
- · lineSegment.cpp

8.11 Player Class Reference

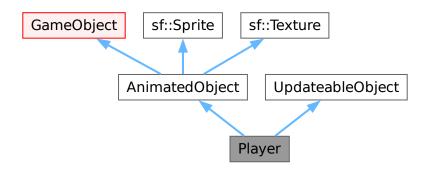
Player class.

```
#include <player.hpp>
```

Inheritance diagram for Player:



Collaboration diagram for Player:



Public Types

enum class MoveDirection {
 north , east , south , west ,
 count }

Directions of movement.

Public Member Functions

virtual ~Player ()

Destructor.

- virtual void setIsMoving (MoveDirection direction)
- · virtual void stopMoving (MoveDirection direction)
- void stopMoving ()

Stops player from moving.

• bool isMoving () const

Checks if player is moving.

• bool isMoving (MoveDirection direction) const

Checks if player is moving in a specific direction.

void setMovementSpeed (float speed)

Player movement speed setter.

virtual float getMovementSpeed () const

Player movement speed getter.

bool isMovingDiagonaly () const

Checks if player is moving diagonally.

Point2d getMoveVectorOrigin () const

Returns a 2D vector containing information on which cardinal directions the player is moving.

Point2d getMoveVector () const

Returns the player's movement vector multiplied by speed.

virtual void update ()

Updates state of object.

· virtual void setPosition (Point2d pos)

Object position setter.

• virtual void animate ()

Animates object.

• virtual Point2d getPosition () const

Object position getter.

Private Member Functions

• void move (Point2d vec)

Moves player on screen.

Private Attributes

• float m_movementSpeed {50}

Player movement speed.

std::array< bool, static_cast< ssize_t >(MoveDirection::count)> m_isMoving {}

Array with player's current movement direction.

• Point2d m_position {}

Position of object on screen.

8.11.1 Detailed Description

Player class.

Definition at line 16 of file player.hpp.

8.11.2 Member Enumeration Documentation

8.11.2.1 MoveDirection

```
enum class Player::MoveDirection [strong]
```

Directions of movement.

Enumerator

north	
east	
south	
west	
count	

Definition at line 21 of file player.hpp.

00021 { north, east, south, west, count };

8.11.3 Constructor & Destructor Documentation

8.11.3.1 \sim Player()

```
Player::~Player ( ) [virtual]
```

Destructor.

Definition at line 9 of file player.cpp.

00009 {};

8.11.4 Member Function Documentation

8.11.4.1 animate()

```
void AnimatedObject::animate ( ) [virtual], [inherited]
```

Animates object.

Reimplemented in Bush, AnimatedSpriteSheet, FidgetSpinner, and GamePlayer.

Definition at line 5 of file animatedObject.cpp.

References LOGWARN.

Referenced by Engine::animateObjects().

```
Engine::render Engine::animateObjects AnimatedObject::animate
```

8.11.4.2 getMovementSpeed()

```
float Player::getMovementSpeed ( ) const [virtual]
```

Player movement speed getter.

```
Definition at line 32 of file player.cpp. 00032 { return m_movementSpeed; };
```

References m_movementSpeed.

8.11.4.3 getMoveVector()

```
Point2d Player::getMoveVector ( ) const
```

Returns the player's movement vector multiplied by speed.

References Engine::getInstance(), Engine::getLastFrameDuration(), getMoveVectorOrigin(), and m_movementSpeed.

Referenced by move(), and update().

Here is the call graph for this function:





8.11.4.4 getMoveVectorOrigin()

```
Point2d Player::getMoveVectorOrigin ( ) const
```

Returns a 2D vector containing information on which cardinal directions the player is moving.

if (m_isMoving[static_cast<int>(MoveDirection::west)])

References east, m_isMoving, north, south, and west.

Referenced by getMoveVector(), isMoving(), isMovingDiagonaly(), and throwBallPlayer().

Here is the caller graph for this function:

 $vec += \{-1, 0\};$

return vec;

00026 00027

00028

00029 }



8.11.4.5 getPosition()

```
Point2d GameObject::getPosition ( ) const [virtual], [inherited]
```

Object position getter.

Definition at line 7 of file GameObject.cpp. 00007 { return m_position; }

References GameObject::m_position.

8.11.4.6 isMoving() [1/2]

```
bool Player::isMoving ( ) const
```

Checks if player is moving.

```
Definition at line 53 of file player.cpp.
```

```
00053
00054 bool moving{};
00055 return getMoveVectorOrigin() != Point2d{0, 0};
00056 }
```

References getMoveVectorOrigin().

Referenced by GamePlayer::update().

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.4.7 isMoving() [2/2]

Checks if player is moving in a specific direction.

Parameters

```
direction Direction checked.
```

Definition at line 58 of file player.cpp.

References m is Moving.

8.11.4.8 isMovingDiagonaly()

```
bool Player::isMovingDiagonaly ( ) const
```

Checks if player is moving diagonally.

Definition at line 34 of file player.cpp.

References getMoveVectorOrigin().

Referenced by GamePlayer::update().

Here is the call graph for this function:



Here is the caller graph for this function:

```
handlePlayerMovement GamePlayer::update Player::isMovingDiagonaly
```

8.11.4.9 move()

Moves player on screen.

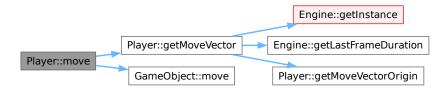
Parameters

```
vec Vector by which player is moved.
```

Reimplemented from AnimatedObject.

References getMoveVector(), and GameObject::move().

Referenced by update().



Here is the caller graph for this function:



8.11.4.10 setIsMoving()

References m_isMoving.

Referenced by keyPressedEventHandler().

Here is the caller graph for this function:



8.11.4.11 setMovementSpeed()

Player movement speed setter.

Definition at line 31 of file player.cpp. 00031 { m_movementSpeed = speed; }

References m_movementSpeed.

Referenced by initialziePlayer().

Here is the caller graph for this function:



8.11.4.12 setPosition()

Object position setter.

Parameters

pos Position to be set.

Reimplemented from GameObject.

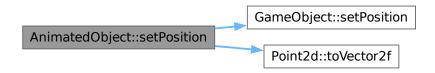
Definition at line 9 of file animatedObject.cpp.

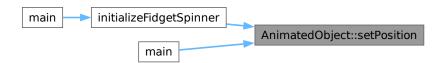
```
00009
00010    GameObject::setPosition(pos);
00011    sf::Sprite::setPosition(pos.toVector2f());
00012 };
```

References GameObject::setPosition(), and Point2d::toVector2f().

Referenced by initializeFidgetSpinner(), and main().

Here is the call graph for this function:





8.11.4.13 stopMoving() [1/2]

```
void Player::stopMoving ( )
```

Stops player from moving.

Definition at line 62 of file player.cpp.

References m_isMoving.

8.11.4.14 stopMoving() [2/2]

Parameters

direction

Definition at line 49 of file player.cpp.

```
00049
00050 m_isMoving[static_cast<ssize_t>(direction)] = false;
00051 }
```

References m_isMoving.

Referenced by keyReleasedEventHandler().

Here is the caller graph for this function:



8.11.4.15 update()

```
void Player::update ( ) [virtual]
```

Updates state of object.

Implements UpdateableObject.

Reimplemented in GamePlayer.

```
Definition at line 68 of file player.cpp.
```

```
00068 {
00069 Point2d vec = getMoveVector();
```

```
00070 move(vec);
00071 };
```

References getMoveVector(), and move().

Referenced by GamePlayer::update().

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5 Member Data Documentation

8.11.5.1 m_isMoving

```
std::array<bool, static_cast<ssize_t>(MoveDirection::count)> Player::m_isMoving {} [private]
```

Array with player's current movement direction.

Definition at line 31 of file player.hpp. $00031 \ {}$;

Referenced by getMoveVectorOrigin(), isMoving(), setIsMoving(), stopMoving(), and stopMoving().

8.11.5.2 m_movementSpeed

```
float Player::m_movementSpeed {50} [private]
```

Player movement speed.

Definition at line 27 of file player.hpp. 00027 {50};

Referenced by getMovementSpeed(), getMoveVector(), and setMovementSpeed().

8.11.5.3 m_position

```
Point2d GameObject::m_position {} [private], [inherited]
```

Position of object on screen.

Definition at line 14 of file GameObject.hpp. 00014 {};

Referenced by GameObject::getPosition(), GameObject::move(), and GameObject::setPosition().

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

- player.hpp
- · player.cpp

8.12 Point2d Class Reference

Class containing 2D point co-ordinates.

```
#include <point2d.hpp>
```

Public Types

• using cordinate_t = float

Public Member Functions

• Point2d (cordinate_t x=0, cordinate_t y=0)

Constructor.

• Point2d (const Point2d &point)=default

Copy constructor.

Point2d (const sf::Vector2f &vector)

Constructor.

- Point2d (const sf::Vector2i &vector)
- cordinate_t getX () const

X co-ordinate getter.

cordinate_t getY () const

Y co-ordinate getter.

void setX (cordinate_t x)

X co-ordinate setter.

void setY (cordinate_t y)

Y co-ordinate setter.

void swap (Point2d &b)

Swaps co-ordinates with another point.

• sf::Vector2f toVector2f () const

Convert to Vector2f.

cordinate_t length () const

Get length of vector.

cordinate_t distanceTo (const Point2d &point) const

Get distance between two points.

Private Attributes

```
cordinate_t m_x
```

X co-ordinate of the point.

cordinate_t m_y

Y co-ordinate of the point.

Friends

- Point2d operator+ (const Point2d &a, const Point2d &b)
- Point2d & operator+= (Point2d &a, const Point2d &b)
- Point2d operator- (const Point2d &a, const Point2d &b)
- bool operator== (const Point2d &a, const Point2d &b)
- bool operator!= (const Point2d &a, const Point2d &b)
- Point2d operator* (const Point2d &a, float b)
- Point2d operator* (float b, const Point2d &a)
- std::ostream & operator<< (std::ostream &os, const Point2d &point)
- std::istream & operator>> (std::istream &is, Point2d &point)

8.12.1 Detailed Description

Class containing 2D point co-ordinates.

Definition at line 9 of file point2d.hpp.

8.12.2 Member Typedef Documentation

8.12.2.1 cordinate_t

```
using Point2d::cordinate_t = float
```

Definition at line 11 of file point2d.hpp.

8.12.3 Constructor & Destructor Documentation

8.12.3.1 Point2d() [1/4]

Constructor.

Parameters

X	X co-ordinate of point.
У	Y co-ordinate of point.

```
Definition at line 10 of file point2d.cpp.
```

```
00010 : m_x(x), m_y(y) {};
```

8.12.3.2 Point2d() [2/4]

Copy constructor.

Parameters

	point	Instance of class Point2d.	
--	-------	----------------------------	--

8.12.3.3 Point2d() [3/4]

Constructor.

Parameters

```
vector 2D vector.
```

Definition at line 11 of file point2d.cpp.

```
00011 : m_x{vector.x}, m_y{vector.y} {};
```

8.12.3.4 Point2d() [4/4]

Definition at line 12 of file point2d.cpp.

```
00013 : m_x{static_cast<cordinate_t>(vector.x)},
00014 m_y{static_cast<cordinate_t>(vector.y)} {};
```

8.12.4 Member Function Documentation

8.12.4.1 distanceTo()

Get distance between two points.

Parameters

```
Definition at line 77 of file point2d.cpp.
```

```
00077
00078    return (*this - point).length();
00079 };
```

8.12.4.2 getX()

```
Point2d::cordinate_t Point2d::getX ( ) const
```

X co-ordinate getter.

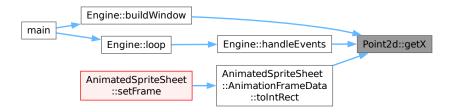
Definition at line 54 of file point2d.cpp.

```
00054 { return m_x; };
```

References m_x.

Referenced by Engine::buildWindow(), Engine::handleEvents(), and AnimatedSpriteSheet::AnimationFrameData::toIntRect().

Here is the caller graph for this function:



8.12.4.3 getY()

```
Point2d::cordinate_t Point2d::getY ( ) const
```

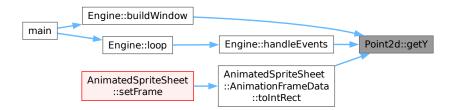
Y co-ordinate getter.

Definition at line 55 of file point2d.cpp.

```
00055 { return m_y; };
```

References m_y.

Referenced by Engine::buildWindow(), Engine::handleEvents(), and AnimatedSpriteSheet::AnimationFrameData::toIntRect().



8.12.4.4 length()

```
Point2d::cordinate_t Point2d::length ( ) const
```

Get length of vector.

Definition at line 72 of file point2d.cpp.

```
00072
00073    return std::sqrt((m_x * m_x) + (m_y * m_y));
00074    return length();
00075 }
```

References length(), m_x, and m_y.

Referenced by length().

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.4.5 setX()

X co-ordinate setter.

Definition at line 57 of file point2d.cpp. 00057 { $m_x = x$; };

References m_x.

Referenced by Engine::handleEvents().

Here is the caller graph for this function:



8.12.4.6 setY()

Y co-ordinate setter.

Definition at line 58 of file point2d.cpp. 00058 { $m_y = y$; };

References m_y.

Referenced by Engine::handleEvents().

Here is the caller graph for this function:



8.12.4.7 swap()

Swaps co-ordinates with another point.

Parameters

b | Point to swap co-ordinates with.

```
Definition at line 63 of file point2d.cpp.
```

```
00063
00064 std::swap(m_x, b.m_x);
00065 std::swap(m_y, b.m_y);
00066 }
```

References m_x, and m_y.

8.12.4.8 toVector2f()

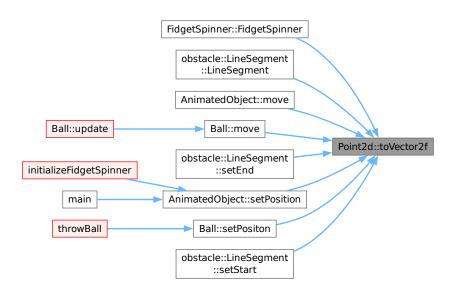
```
sf::Vector2f Point2d::toVector2f ( ) const
```

Convert to Vector2f.

References m_x, and m_y.

Referenced by FidgetSpinner::FidgetSpinner(), obstacle::LineSegment::LineSegment(), AnimatedObject::move(), Ball::move(), obstacle::LineSegment::setEnd(), AnimatedObject::setPosition(), Ball::setPosition(), and obstacle::LineSegment::setStart

Here is the caller graph for this function:



8.12.5 Friends And Related Symbol Documentation

8.12.5.1 operator"!=

{

{

```
8.12.5.2 operator* [1/2]
```

```
Point2d operator* (
            const Point2d & a,
             float b) [friend]
Definition at line 29 of file point2d.cpp.
00029
       return {a.getX() * b, a.getY() * b};
00031 };
8.12.5.3 operator* [2/2]
Point2d operator* (
              const Point2d & a ) [friend]
Definition at line 33 of file point2d.cpp.
00033 { return operator*(a, b); }
8.12.5.4 operator+
Point2d operator+ (
             const Point2d & a,
              const Point2d & b ) [friend]
Definition at line 16 of file point2d.cpp.
00017
       return {a.m_x + b.m_x, a.m_y + b.m_y};
00018 }
8.12.5.5 operator+=
Point2d & operator+= (
            Point2d & a,
             const Point2d & b ) [friend]
Definition at line 20 of file point2d.cpp.
00020
                                                      {
00021
       a = a + b;
00022 return a;
00023 }
8.12.5.6 operator-
Point2d operator- (
             const Point2d & a,
              const Point2d & b ) [friend]
```

Definition at line 25 of file point2d.cpp.

return operator+(a, -1 * b);

00025

00026

00027 }

std::ostream & operator<< (</pre>

8.12.5.7 operator <<

8.12.5.8 operator==

Definition at line 46 of file point2d.cpp.

```
00046

00047 return a.getX() == b.getX() && a.getY() == b.getY();

00048 }
```

8.12.5.9 operator>>

Definition at line 40 of file point2d.cpp.

```
00041 is » point.m_x;
00042 is » point.m_y;
00043 return is;
00044 }
```

8.12.6 Member Data Documentation

8.12.6.1 m x

```
cordinate_t Point2d::m_x [private]
```

X co-ordinate of the point.

Definition at line 17 of file point2d.hpp.

Referenced by getX(), length(), setX(), swap(), and toVector2f().

8.12.6.2 m_y

```
cordinate_t Point2d::m_y [private]
```

Y co-ordinate of the point.

Definition at line 21 of file point2d.hpp.

Referenced by getY(), length(), setY(), swap(), and toVector2f().

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

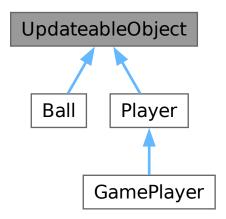
- point2d.hpp
- point2d.cpp

8.13 UpdateableObject Class Reference

Updateable object class.

#include <updateableObject.hpp>

Inheritance diagram for UpdateableObject:



Public Member Functions

virtual void update ()=0
 Updates state of object.

8.13.1 Detailed Description

Updateable object class.

Definition at line 6 of file updateableObject.hpp.

8.13.2 Member Function Documentation

8.13.2.1 update()

virtual void UpdateableObject::update () [pure virtual]

Updates state of object.

Implemented in Player, Ball, and GamePlayer.

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

• updateableObject.hpp

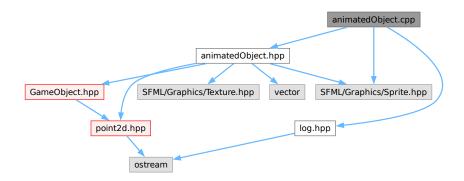
Chapter 9

File Documentation

9.1 README.md File Reference

9.2 animatedObject.cpp File Reference

```
#include "animatedObject.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "log.hpp"
Include dependency graph for animatedObject.cpp:
```



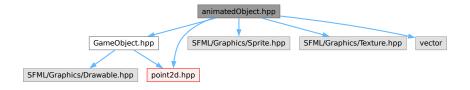
9.3 animatedObject.cpp

```
00001 #include "animatedObject.hpp"
00002 #include "SFML/Graphics/Sprite.hpp"
00003 #include "log.hpp"
00004
00005 void AnimatedObject::animate() {
00006 LOGWARN « "Not overloaded " « this « '\n';
00007 };
80000
00009 void AnimatedObject::setPosition(Point2d pos) {
00010 GameObject::setPosition(pos);
00011 sf::Sprite::setPosition(pos.to
         sf::Sprite::setPosition(pos.toVector2f());
00012 };
00013
00014 void AnimatedObject::move(Point2d vec) {
00015 GameObject::move(vec);
         sf::Sprite::move(vec.toVector2f());
00016
00017
         // LOGINFO « GameObject::getPosition() « '\n';
```

9.4 animatedObject.hpp File Reference

```
#include "GameObject.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "SFML/Graphics/Texture.hpp"
#include "point2d.hpp"
#include <vector>
```

Include dependency graph for animatedObject.hpp:



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



Classes

class AnimatedObject

Animated object class.

9.5 animatedObject.hpp

```
00001 #ifndef ANIMATEDOBJECT H
00002 #define ANIMATEDOBJECT_H_
00004 #include "GameObject.hpp"
00005 #include "SFML/Graphics/Sprite.hpp"
00006 #include "SFML/Graphics/Sprite.npp"
00007 #include "point2d.hpp"
00008 #include <vector>
00009
00013 class AnimatedObject : public GameObject,
00014
                                  public sf::Sprite,
00015
                                  public sf::Texture {
00016
00017 public:
         virtual ~AnimatedObject(){};
00026
         virtual void setPosition(Point2d pos);
00030
        virtual void animate();
00031
00032
         virtual void move (Point2d vec);
00033
00034 private:
00035 };
00036
00037 #endif // ANIMATEDOBJECT_H_
```

9.6 animatedSpriteSheet.cpp File Reference

```
#include "animatedSpriteSheet.hpp"
#include "engine.hpp"
#include "log.hpp"
#include <fstream>
#include <functional>
#include <iostream>
#include <limits>
#include <string>
#include <string_view>
```

Include dependency graph for animatedSpriteSheet.cpp:



Functions

std::istream & operator>> (std::istream &is, AnimatedSpriteSheet::AnimationFrameData &afd)

9.6.1 Function Documentation

9.6.1.1 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            AnimatedSpriteSheet::AnimationFrameData & afd )
```

Definition at line 11 of file animatedSpriteSheet.cpp.

```
00012
00013
         LOGTRACEN;
         is » afd.m_position;
00014
        is » afd.m_size;
is » afd.m_duration;
00015
00016
        return is;
00017
00018 }
```

9.7 animatedSpriteSheet.cpp

```
00001 #include "animatedSpriteSheet.hpp"
00002 #include "engine.hpp"
00003 #include "log.hpp"
00004 #include <fstream>
00005 #include <functional>
00006 #include <iostream>
00007 #include <limits>
00008 #include <string>
00009 #include <string_view>
00010
00011 std::istream &operator»(std::istream &is,
00012
                                   AnimatedSpriteSheet::AnimationFrameData &afd) {
00013
        LOGTRACEN:
00014 is » afd.m_position;
00015 is » afd.m_size;
        is » afd.m_duration;
```

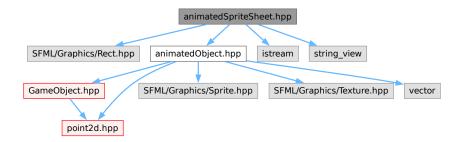
```
00017
       return is;
00018 }
00019
00020 sf::IntRect AnimatedSpriteSheet::AnimationFrameData::toIntRect() const {
00021
       LOGTRACEN:
00022
        return {static cast<int>(m position.getX()),
                static_cast<int>(m_position.getY()), static_cast<int>(m_size.getX()),
00024
                static_cast<int>(m_size.getY())};
00025 }
00026
00027 AnimatedSpriteSheet::AnimatedSpriteSheet(std::string_view path) {
        std::string pathh{std::string(path) + "/spritesheet.png"};
00028
00029
        LOGINFO « pathh « '
                           '\n';
00030
       loadFromConfigFile(path);
00031
        setTexture(*this);
00032
        setFrame(getCurrentAnimationFrameData());
00033
00034
       // setFrame(getCurrentAnimationFrameData());
00035 };
00036
00037 void AnimatedSpriteSheet::animate() {
00038
       LOGTRACEN;
00039
        m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
       // if duration of frame not exhaused do nothing;
00040
00041
        if (m_animationTimer < getCurrentAnimationFrameDuration())</pre>
00042
          return;
00043
        nextFrame();
00044 }
00045
00046 void AnimatedSpriteSheet::nextFrame() {
00047 LOGTRACEN;
00048
       // change frame
00049
        ++m_currentFrameId;
00050
        m_currentFrameId %= getCurrentAnimationFrameCount();
00051
        setFrame(getCurrentAnimationFrameData());
00052 }
00053
00054 int AnimatedSpriteSheet::getCurrentAnimationFrameCount() const {
00055
       LOGTRACEN:
00056
        return m_animationsData[m_currentAnimationTypeIndex].size();
00057 }
00058
00059 const AnimatedSpriteSheet::AnimationFrameData &
00060 AnimatedSpriteSheet::getCurrentAnimationFrameData() const {
00061
       LOGTRACEN;
00062
        if (m_animationsData.size() == 0 ||
00063
            m_animationsData.size() < m_currentAnimationTypeIndex) {</pre>
00064
         LOGWARN « "No animation data\n";
00065
       } else if (m_animationsData[m_currentAnimationTypeIndex].size() == 0 ||
00066
                   m_animationsData[m_currentAnimationTypeIndex].size() <</pre>
00067
                       m_currentAnimationTypeIndex) {
          LOGWARN « "No frame data in animation " « m_currentAnimationTypeIndex
00068
                  « '\n';
00069
00070
00071
        return m_animationsData[m_currentAnimationTypeIndex][m_currentFrameId];
00072 }
00073
00074 float AnimatedSpriteSheet::getCurrentAnimationFrameDuration() const {
00075 LOGTRACEN;
00076
        return getCurrentAnimationFrameData().m_duration;
00077 }:
00078
00079 void AnimatedSpriteSheet::setFrame(const AnimationFrameData &frameData) {
08000
     LOGTRACEN;
00081
        setTextureRect(frameData.toIntRect());
       m_animationTimer = 0;
00082
00083 }
00084
00085 void AnimatedSpriteSheet::loadFrame(std::istream &stream,
00086
                                           animationData_t &animationData) {
00087
        LOGTRACEN:
00088
        // load Frame data
00089
        AnimationFrameData fdat{};
00090
        stream » fdat:
       // LOGINFO « fdat.m_position « " " « fdat.m_size « " " « // fdat.m_duration
00091
00092
00093
                   « '\n';
00094
00095
        // add frame data to animation
00096
        animationData.push_back(fdat);
        stream.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00097
00098 };
00099
00100 void AnimatedSpriteSheet::loadSpritesheet(
        std::istream &stream, std::string_view configFileDirectoryPath) {
LOGINFO « "loading spritesheet data\n";
00101
00102
       std::string spriteSheetRelPath{};
00103
```

```
00104
        stream » spriteSheetRelPath;
        spriteSheetRelPath =
00105
             std::string(configFileDirectoryPath) + "/" + spriteSheetRelPath;
00106
00107
00108
        loadFromFile(spriteSheetRelPath);
00109
00110
        stream.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00111 }
00112
00113 void AnimatedSpriteSheet::loadFromConfigFile(std::string_view pathToDir) {
        std::string configFilePath{static_cast<std::string>(pathToDir) + "/config.txt"};
00114
00115
00116
        // open file
00117
        std::fstream configFile{configFilePath};
00118
        if (!configFile.is_open()) {
00119
         LOGERROR « "opening config file failed it should be at " « configFilePath
                    « "\n";
00120
00121
          return;
00122
00123
00124
        animationData_t *animationDataBeingLoaded{NULL};
00125
        std::string line{};
00126
00127
        while (!configFile.eof()) {
00128
         line = "";
00129
          // load line, ignore lines that are commnets
00130
          std::getline(configFile, line);
} while (line.starts_with("#"));
00131
00132
00133
00134
          // load spritesheet
00135
          if (line.starts_with("SPRITESHEET"))
00136
             loadSpritesheet(configFile, pathToDir);
00137
          } else if (line.starts_with("ANIMATION")) {
           // Add new animation and change pointer
// LOGINFO « "loading animation data\n";
00138
00139
            m_animationsData.push_back({});
00140
            animationDataBeingLoaded = {&m_animationsData.back()};
00142
00143
           // load frame
00144
          else if (line.starts_with("FRAME")) {
            if (animationDataBeingLoaded == NULL)
00145
               LOGERROR « "config file is corrupted, FRAME before ANIMATION?";
00146
00147
             loadFrame(configFile, *animationDataBeingLoaded);
00148
           // not known
00149
          else {
   // empty line ignore
   if (line == "") {
00150
00151
00152
00153
              continue;
00154
00155
             // invalid line
             LOGWARN « "not recegnized config option: " « line « " in file: " « configFilePath « '\n';
00156
00157
00158
          }
00159
        // close file
00161
        configFile.close();
00162 }
```

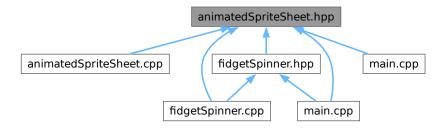
9.8 animatedSpriteSheet.hpp File Reference

```
#include "SFML/Graphics/Rect.hpp"
#include "animatedObject.hpp"
#include <istream>
#include <string_view>
```

Include dependency graph for animatedSpriteSheet.hpp:



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



Classes

- · class AnimatedSpriteSheet
- · struct AnimatedSpriteSheet::AnimationFrameData

9.9 animatedSpriteSheet.hpp

```
00001 #ifndef ANIMATEDSPRITESHEET_H_
00002 #define ANIMATEDSPRITESHEET_H_
00003
00004 #include "SFML/Graphics/Rect.hpp"
00005 #include "animatedObject.hpp"
00006 #include <istream>
00007 #include <string_view>
00008 class AnimatedSpriteSheet : public AnimatedObject {
00009
00010 public:
        struct AnimationFrameData {
00012
          //** Duration of frame */
00013
           float m_duration;
00014
           //** size of sprite */
00015
           Point2d m_size;
00016
           //** position of sprite */
00017
           Point2d m_position;
00018
00019
           friend std::istream &operator»(std::istream &is, AnimationFrameData &afd);
00020
00021
           sf::IntRect toIntRect() const;
00022
00023
```

```
using animationData_t = std::vector<AnimationFrameData>;
00025
00026 public:
00027
       AnimatedSpriteSheet(std::string_view path);
00028
00029
       virtual void animate() override;
00031 protected:
       int getCurrentAnimationFrameCount() const;
00032
00033
       sf::IntRect getCurrentAnimationFrameRect() const;
00034
       float getCurrentAnimationFrameDuration() const;
00035
       const AnimationFrameData &getCurrentAnimationFrameData() const;
00036
00037 protected:
00038
       void nextFrame();
00039
       void setFrame(const AnimationFrameData &frameData);
00040
00044
       void loadFrame(std::istream &stream, animationData t &animationData);
00045
00046
       void loadSpritesheet(std::istream &stream,
00047
                             std::string_view configFileDirectoryPath);
00048
00049
       void loadFromConfigFile(std::string_view pathToDir);
00050
00051 private:
       std::vector<animationData_t> m_animationsData;
00056
00060
       int m_currentAnimationTypeIndex;
00061
       int m_currentFrameId{};
00062
00063
       float m animationTimer{};
00064 };
00066 #endif // ANIMATEDSPRITESHEET_H_
```

9.10 engine.cpp File Reference

```
#include "engine.hpp"
#include "SFML/Graphics/Drawable.hpp"
#include "SFML/Graphics/Rect.hpp"
#include "SFML/Graphics/View.hpp"
#include "SFML/System/Time.hpp"
#include "SFML/Window/Event.hpp"
#include "SFML/Window/VideoMode.hpp"
#include "log.hpp"
#include <algorithm>
#include <iostream>
```

Include dependency graph for engine.cpp:



9.11 engine.cpp

```
00001 #include "engine.hpp"
00002 #include "SFML/Graphics/Drawable.hpp"
00003 #include "SFML/Graphics/Rect.hpp"
00004 #include "SFML/Graphics/View.hpp"
00005 #include "SFML/System/Time.hpp"
00006 #include "SFML/Window/Event.hpp"
00007 #include "SFML/Window/VideoMode.hpp"
00008 #include "log.hpp"
00009 #include "algorithm>
```

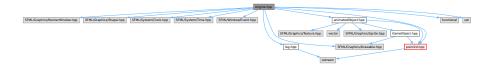
```
00010 #include <iostream>
00011
00012 Engine *Engine::s_instancePtr{nullptr};
00013
00014 Engine::Engine() {
00015
        LOGINFON;
       m_eventHandlers.fill([](const sf::Event &event) {});
00017
00018 }
00019
00020 Engine::~Engine() {
00021
        LOGINFON:
00022
        for (sf::Drawable *it : m drawablesCollection) {
00023
         delete it;
00024
00025
       for (AnimatedObject *it : m_animatedObjectsCollection) {
00026
         delete it;
       }
00027
00028 }
00029
00030 Engine &Engine::getInstance() {
00031
       if (s_instancePtr == nullptr)
00032
         s_instancePtr = new Engine;
00033
       return *s_instancePtr;
00034 }
00035
00036 Engine &Engine::setMaxFps(int fps) {
00037
        LOGINFON;
00038
       m_maxFPS = fps;
00039
       m_window.setFramerateLimit(fps);
00040
       return getInstance();
00041 }
00042
00043 Engine &Engine::setWindowTitle(std::string_view title) {
00044
       m_windowTitle = title;
00045
        return getInstance();
00046 }
00047
00048 Engine &Engine::setResolution(Point2d resolution) {
00049
       m_resoltuon = resolution;
00050
       return getInstance();
00051 }
00052
00053 Engine &Engine::buildWindow() {
00054
        LOGINFON;
00055
       m_window.create({static_cast<unsigned int>(m_resoltuon.getX()),
00056
                         static_cast<unsigned int>(m_resoltuon.getY())},
00057
                        m windowTitle);
00058
       m_window.setFramerateLimit(m_maxFPS);
00059
       return getInstance();
00060 }
00061
00062 Engine &Engine::setEventHandler(sf::Event::EventType eventType,
00063
                                      eventHandler_t handler) {
00064
        m_eventHandlers[eventType] = handler;
00065
       return *this:
00066 }
00067
00068 Point2d Engine::getResolution() const { return m_resoltuon; }
00069
00070 void Engine::handleEvents() {
00071 sf::Event event;
       while (m_window.pollEvent(event)) {
00072
00073
        if (event.type == sf::Event::Closed)
00074
            m_window.close();
00075
         if (event.type == sf::Event::Resized) {
00076
00077
            // Update engine info
00078
           m resoltuon.setX(event.size.width);
00079
           m_resoltuon.setY(event.size.height);
00080
            // make new view
00081
            sf::FloatRect view{0, 0, static_cast<float>(m_resoltuon.getX()),
00082
                               static_cast<float>(m_resoltuon.getY())};
00083
            m_window.setView(sf::View{view});
00084
00085
            LOGINFO « "Resized\t" « event.size.width « '\t' « event.size.height
                   « '\n';
00086
00087
00088
00089
          // Fire custom event handler.
00090
         m eventHandlers[event.type](event);
00091
00092 }
00093
00094 void Engine::clear() { m_window.clear(); }
00095
00096 void Engine::render() {
00097
       animateObjects();
```

```
00098
       drawDrawables();
00099 }
00100
00101 void Engine::display() { m_window.display(); }
00102
00103 void Engine::loop() {
00104 LOGINFON;
00105
00106
       while (m_window.isOpen()) {
00107
00108
         handleEvents();
00109
00110
          // restart clock
00111
         m_lastFrameDuration = m_clock.restart();
00112
00113
          m_loopFunction();
00114
00115
          render();
00116
          display();
00117
       }
00118 }
00119
00120 Engine::RenderWindow & Engine::getWindow() { return m_window; }
00121
00122 void Engine::add(Drawable *drawable) {
00123 LOGINFO « drawable « '\n';
       m_drawablesCollection.insert(drawable);
00124
00125 }
00126
00127 void Engine::add(AnimatedObject *animatedObject) {
00128 LOGINFO « animatedObject « '\n';
00129
       m_animatedObjectsCollection.insert(animatedObject);
00130 }
00131
00132 void Engine::remove(Engine::Drawable *drawable) {
00133
       m_drawablesCollection.erase(drawable);
00134 }
00136 Engine::Time Engine::getLastFrameDuration() const {
00137
       return m_lastFrameDuration;
00138 }
00139
00140 void Engine::drawDrawables() {
00141 for (auto &it: m_drawablesCollection)
00142 getWindow().draw(*it);
00143 }
00144
00145 void Engine::animateObjects() {
00146 for (auto it : m_animatedObjectsCollection) {
00147 it->animate();
         getWindow().draw(*it);
00149 }
00150 }
```

9.12 engine.hpp File Reference

```
#include "SFML/Graphics/RenderWindow.hpp"
#include "SFML/Graphics/Shape.hpp"
#include "SFML/System/Clock.hpp"
#include "SFML/System/Time.hpp"
#include "SFML/Window/Event.hpp"
#include "animatedObject.hpp"
#include "log.hpp"
#include "point2d.hpp"
#include <SFML/Graphics/Drawable.hpp>
#include <functional>
#include <set>
```

Include dependency graph for engine.hpp:



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



Classes

· class Engine

The engine class.

9.13 engine.hpp

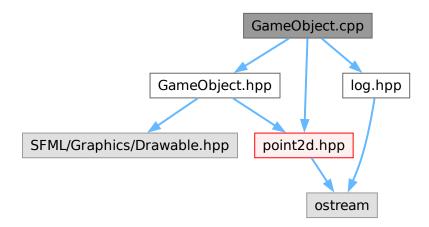
```
00001 #ifndef ENGINE_HPP_
00002 #define ENGINE_HPP_
00003
00004 #include "SFML/Graphics/RenderWindow.hpp"
00005 #include "SFML/Graphics/Shape.hpp"
00006 #include "SFML/System/Clock.hpp
00000 #include "SFML/System/Time.hpp"
00008 #include "SFML/Window/Event.hpp"
00008 #include "srmi/window/Event.n
00009 #include "animatedObject.hpp"
00010 #include "log.hpp"
00011 #include "point2d.hpp"
00012 #include <SFML/Graphics/Drawable.hpp>
00013 #include <functional>
00014 #include <set>
00015
00018 class Engine {
00019 public:
00020
        using Time = sf::Time;
00021
        using Clock = sf::Clock;
00022
        using RenderWindow = sf::RenderWindow;
00023
        using Event = sf::Event;
00024
        using Drawable = sf::Drawable;
00025
        using Shape = sf::Shape;
00026
00027
        using eventHandler_t = std::function<void(const Event &)>;
00028
        using drawableCollection_t = std::set<Drawable *>;
00029
        using animatedObjecsCollection_t = std::set<AnimatedObject *>;
00030
00031 private:
00034
        static Engine *s instancePtr;
00035
00038
        std::function<void()> m_loopFunction{[]() {}};
00039
00042
        RenderWindow m_window{};
00045
        Point2d m_resoltuon{1000, 800};
std::string m_windowTitle{"dev"};
00048
00052
        int m_maxFPS{60};
00053
00057
         std::array<eventHandler_t, Event::Count> m_eventHandlers;
00058
00062
        drawableCollection t m drawablesCollection{};
00063
00064
        animatedObjecsCollection_t m_animatedObjectsCollection{};
00065
```

```
Clock m_clock{};
00069
00073
        Time m_lastFrameDuration{};
00074
00075 public:
00079
        Engine():
        ~Engine();
00084
00087
        static Engine &getInstance();
00088
00092
       Engine &setMaxFps(int fps);
00093
00097
        Engine &setWindowTitle(std::string_view title);
00098
00102
        Engine &setResolution(Point2d resolution);
00103
        Engine &setResolution(Point2d::cordinate_t x, Point2d::cordinate_t y) {
00106
00107
         setResolution({x, y});
00108
          return *this;
00109
00110
00114
        Engine &setLoopFunction(std::function<void()> function) {
00115
        m_loopFunction = function;
00116
          return *this;
00117
00118
00124
        Engine &setEventHandler(Event::EventType eventType, eventHandler_t handler);
00125
00128
        Point2d getResolution() const;
00129
00132
       Engine &buildWindow();
00133
00137
       void handleEvents();
00138
00142
       void clear();
00143
00147
       void render();
00152
       void display();
00153
00157
       void loop();
00158
       RenderWindow &getWindow():
00162
00163
00168
        void add(Drawable *drawable);
00173
        void add(AnimatedObject *animatedObject);
00174
00175
       void remove(Drawable *);
00176
00180
       Time getLastFrameDuration() const;
00181
00182 private:
00186
       void drawDrawables();
00190
       void animateObjects();
00191 };
00192
00193 #endif // ENGINE_HPP_
```

9.14 GameObject.cpp File Reference

```
#include "GameObject.hpp"
#include "log.hpp"
#include "point2d.hpp"
```

Include dependency graph for GameObject.cpp:

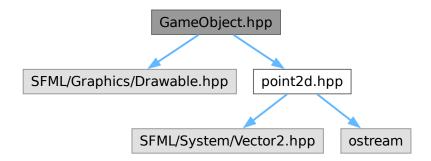


9.15 GameObject.cpp

Go to the documentation of this file.

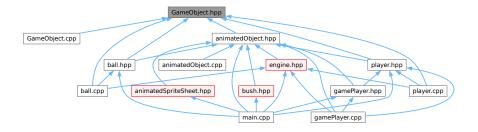
9.16 GameObject.hpp File Reference

```
#include "SFML/Graphics/Drawable.hpp"
#include "point2d.hpp"
Include dependency graph for GameObject.hpp:
```



9.17 GameObject.hpp 143

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



Classes

· class GameObject

Game object class.

9.17 GameObject.hpp

Go to the documentation of this file.

```
00001 #ifndef GAMEOBJECT_HPP_
00002 #define GAMEOBJECT_HPP_
00003
00004 #include "SFML/Graphics/Drawable.hpp" 00005 #include "point2d.hpp"
00009 class GameObject {
00010 private:
00014
        Point2d m_position{};
00015
00016 public:
        virtual ~GameObject(){};
        virtual void setPosition(Point2d pos);
00029
        virtual Point2d getPosition() const;
00034
       virtual void move(Point2d vector);
00035 };
00036
00037 #endif // GAMEOBJECT_HPP_
```

9.18 lineSegment.cpp File Reference

```
#include "lineSegment.hpp"
#include "SFML/Graphics/Color.hpp"
#include "SFML/Graphics/PrimitiveType.hpp"
#include "SFML/Graphics/Vertex.hpp"
#include "engine.hpp"
#include "point2d.hpp"
Include dependency graph for lineSegment.cpp:
```



Namespaces

· namespace obstacle

9.18.1 Detailed Description

Deprecated Bad design. Reinventing wheel.

Definition in file lineSegment.cpp.

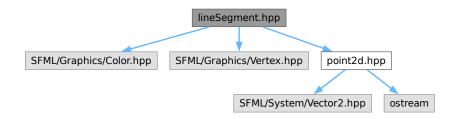
9.19 lineSegment.cpp

```
Go to the documentation of this file.
```

```
00001 #include "lineSegment.hpp"
00002 #include "SFML/Graphics/Color.hpp"
00003 #include "SFML/Graphics/PrimitiveType.hpp"
00004 #include "SFML/Graphics/Vertex.hpp"
00005 #include "engine.hpp"
00006 #include "point2d.hpp"
00007
00013 namespace obstacle {
00014
00015 LineSegment::LineSegment(Point2d start, Point2d end, sf::Color color)
00016
           : m_points{{start.toVector2f(), color}, {end.toVector2f(), color}},
00017
              m color{color} {};
00018
00019 void LineSegment::draw() const {
         Engine::getInstance().getWindow().draw(m_points, 2, sf::Lines);
00021 }
00022
00023 Point2d LineSegment::getStart() const { return Point2d{m_points[0].position}; }
00024 Point2d LineSegment::getEnd() const { return Point2d{m_points[1].position}; }
00026 void LineSegment::setStart(Point2d val) {
00027
        m_points[0].position = val.toVector2f();
00028 }
00029 void LineSegment::setEnd(Point2d val) {
00030
         m_points[1].position = val.toVector2f();
00031 }
00032
00033 }; // namespace obstacle
```

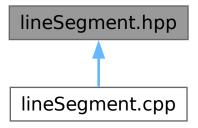
9.20 lineSegment.hpp File Reference

```
#include "SFML/Graphics/Color.hpp"
#include "SFML/Graphics/Vertex.hpp"
#include "point2d.hpp"
Include dependency graph for lineSegment.hpp:
```



9.21 lineSegment.hpp 145

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



Classes

· class obstacle::LineSegment

Namespaces

· namespace obstacle

9.20.1 Detailed Description

Deprecated Bad design. Reinventing wheel.

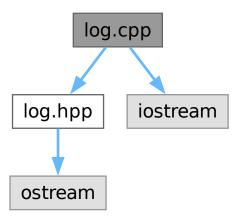
Definition in file lineSegment.hpp.

9.21 lineSegment.hpp

```
00001 #ifndef LINESEGMENT_HPP_
00002 #define LINESEGMENT_HPP_
00004 #include "SFML/Graphics/Color.hpp"
00005 #include "SFML/Graphics/Vertex.hpp"
00006 #include "point2d.hpp"
00007
00013 namespace obstacle {
00014 class LineSegment {
00015    sf::Vertex m_points[2] = {};
00016 sf::Color m_color;
00017
00018 public:
        LineSegment(Point2d start, Point2d end, sf::Color color = {});
00019
00020
00021
        void draw() const;
00022
        Point2d getStart() const;
Point2d getEnd() const;
00023
00024
00025
00026
        void setStart(Point2d val);
00027
         void setEnd(Point2d val);
00028
         void setColor(sf::Color color) { m_color = color; };
00029
00030 };
00031 };
               // namespace obstacle
00032 #endif // LINESEGMENT_HPP_
```

9.22 log.cpp File Reference

```
#include "log.hpp"
#include <iostream>
Include dependency graph for log.cpp:
```



Namespaces

· namespace G

Variables

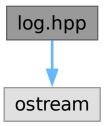
• std::ostream & G::logstream {std::cerr}

9.23 log.cpp

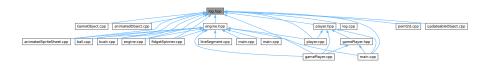
```
00001
00002 #include "log.hpp"
00003
00004 #include <iostream>
00005
00006 namespace G {
00007 std::ostream &logstream{std::cerr};
00008 };
```

9.24 log.hpp File Reference

#include <ostream>
Include dependency graph for log.hpp:



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



Namespaces

· namespace G

Macros

- #define LOGINFON LOGINFO << '\n'
- #define LOGINFO
- #define LOGWARNN LOGWARN << '\n'
- #define LOGWARN
- #define LOGERROR
- #define LOGTRACEN

9.24.1 Macro Definition Documentation

9.24.1.1 LOGERROR

#define LOGERROR

Value:

Definition at line 25 of file log.hpp.

9.24.1.2 LOGINFO

#define LOGINFO

Value:

Definition at line 12 of file log.hpp.

9.24.1.3 LOGINFON

```
#define LOGINFON LOGINFO << '\n'
```

Definition at line 10 of file log.hpp.

9.24.1.4 LOGTRACEN

```
#define LOGTRACEN
```

Definition at line 38 of file log.hpp.

9.24.1.5 LOGWARN

#define LOGWARN

Value:

Definition at line 18 of file log.hpp.

9.24.1.6 LOGWARNN

```
#define LOGWARNN LOGWARN << '\n'
```

Definition at line 16 of file log.hpp.

9.25 log.hpp 149

9.25 log.hpp

Go to the documentation of this file.

```
00001 #ifndef LOG HPP
00002 #define LOG_HPP_
00004 #include <ostream>
00005
00006 namespace G {
00007 extern std::ostream &logstream;
00008 };
00009
00010 #define LOGINFON LOGINFO « '\n'
00011
00012 #define LOGINFO
00016 #define LOGWARNN LOGWARN « ' \ n'
00017
00018 #define LOGWARN
00010 #define bodward

00019 G::logstream « "\033[33m"

00020 « "WARN "
00021
                      "\033[35m"
                        _FILE_NAME__ « " \033[36m" « __PRETTY_FUNCTION__
                   « __FILE_NAM
« "\033[0m "
00023
00024
00025 #define LOGERROR
00026 G::logstream « "\033[31m"
00027
                    « "ERROR "
00028
                      "\033[35m"
00029
                        _FILE_NAME__ « " \033[36m" « __PRETTY_FUNCTION__
                    « "/033[0m "
00030
00031
00032 #ifdef TRACE
00033 #define LOGTRACEN
00034 G::logstream « "TRACE "
              « "\033[35m" « __FILE_NAME__ « " \033[36m"
00036
                    « __PRETTY_FUNCTION__ « "\033[0m\n";
00037 #else
00038 #define LOGTRACEN
00039 #endif // TRACE
00040
00041 #endif // LOG_HPP_
```

9.26 player.cpp File Reference

```
#include "player.hpp"
#include "GameObject.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "engine.hpp"
#include "log.hpp"
#include "point2d.hpp"
#include <bits/ranges_algo.h>
Include dependency graph for player.cpp:
```



9.27 player.cpp

```
00001 #include "player.hpp"
00002 #include "GameObject.hpp"
00003 #include "SFML/Graphics/Sprite.hpp"
00004 #include "engine.hpp"
00005 #include "log.hpp"
00006 #include "point2d.hpp"
00007 #include <bits/ranges_algo.h>
80000
00009 Player::~Player(){};
00010
00011 void Player::move(Point2d vec) {
00012 GameObject::move(vec);
00013
       sf::Sprite::move(getMoveVector().toVector2f());
00014 // LOGINFO « GameObject::getPosition() « '\n';
00015 };
00016
00017 Point2d Player::getMoveVectorOrigin() const {
00018
       Point2d vec{};
00020
       if (m_isMoving[static_cast<int>(MoveDirection::north)])
00021
00022
       if (m_isMoving[static_cast<int>(MoveDirection::east)])
00023
         vec += \{1, 0\};
       if (m_isMoving[static_cast<int>(MoveDirection::south)])
00024
00025
         vec += \{0, 1\};
       if (m_isMoving[static_cast<int>(MoveDirection::west)])
00027
         vec += \{-1, 0\};
00028
       return vec;
00029 }
00030
00031 void Player::setMovementSpeed(float speed) { m_movementSpeed = speed; }
00032 float Player::getMovementSpeed() const { return m_movementSpeed; };
00033
00034 bool Player::isMovingDiagonaly() const
00035 Point2d vec{getMoveVectorOrigin()};
       auto absMoveDist{std::abs(vec.getX()) + std::abs(vec.getY())};
00036
00037
       return absMoveDist > 1;
00039
00040 Point2d Player::getMoveVector() const {
00041 return getMoveVectorOrigin() * m_movementSpeed *
00042
              Engine::getInstance().getLastFrameDuration().asSeconds();
00043 }
00044
00045 void Player::setIsMoving(MoveDirection direction) {
00046
       m_isMoving[static_cast<ssize_t>(direction)] = true;
00047 };
00048
00049 void Player::stopMoving(MoveDirection direction) {
00050
       m isMoving[static cast<ssize t>(direction)] = false;
00051 }
00052
00053 bool Player::isMoving() const {
00054 bool moving{};
       return getMoveVectorOrigin() != Point2d{0, 0};
00055
00056 }
00058 bool Player::isMoving(MoveDirection direction) const {
00059 return m_isMoving[static_cast<ssize_t>(direction)];
00060 }
00061
00062 void Player::stopMoving() {
00063 for (auto &it : m_isMoving) {
         it = false;
00064
00065
       }
00066 }
00067
00068 void Player::update() {
00069 Point2d vec = getMoveVector();
       move(vec);
00071 };
```

9.28 player.hpp File Reference

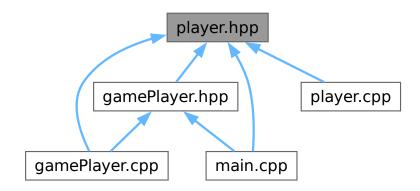
```
#include "GameObject.hpp"
#include "SFML/Graphics/Shape.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "animatedObject.hpp"
#include "log.hpp"
```

9.29 player.hpp 151

```
#include "point2d.hpp"
#include "updateableObject.hpp"
#include <array>
#include <sys/types.h>
Include dependency graph for player.hpp:
```



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



Classes

class Player
 Player class.

9.29 player.hpp

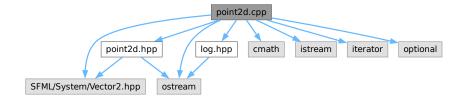
```
00001 #ifndef PLAYER_H_
00002 #define PLAYER_H_
00003
00004 #include "GameObject.hpp"
00005 #include "SFML/Graphics/Shape.hpp"
00006 #include "SFML/Graphics/Sprite.hpp"
00007 #include "animatedObject.hpp"
00008 #include "log.hpp"
00009 #include "point2d.hpp"
00010 #include "updateableObject.hpp"
00011 #include "updateableObject.hpp"
00012 #include <sray>
00012 #include <sys/types.h>
00016 class Player : public AnimatedObject, public UpdateableObject {
00017 public:
00021    enum class MoveDirection { north, east, south, west, count };
00022
```

```
00023 private:
       float m_movementSpeed{50};
00031
        std::array<bool, static_cast<ssize_t>(MoveDirection::count)> m_isMoving{};
00036
       void move(Point2d vec);
00037
00038 public:
       virtual ~Player();
00047
        virtual void setIsMoving(MoveDirection direction);
00052
       virtual void stopMoving(MoveDirection direction);
00053
00057
       void stopMoving();
00061
       bool isMoving() const;
00066
       bool isMoving (MoveDirection direction) const;
00067
00071
       void setMovementSpeed(float speed);
00075
00079
       virtual float getMovementSpeed() const;
       bool isMovingDiagonaly() const;
08000
00084
       Point2d getMoveVectorOrigin() const;
00088
       Point2d getMoveVector() const;
00089
00090
       virtual void update();
00091 };
00092 #endif // PLAYER H
```

9.30 point2d.cpp File Reference

```
#include "point2d.hpp"
#include "SFML/System/Vector2.hpp"
#include "log.hpp"
#include <cmath>
#include <istream>
#include <iterator>
#include <optional>
#include <ostream>
```

Include dependency graph for point2d.cpp:



Functions

- Point2d operator+ (const Point2d &a, const Point2d &b)
- Point2d & operator+= (Point2d &a, const Point2d &b)
- Point2d operator- (const Point2d &a, const Point2d &b)
- Point2d operator* (const Point2d &a, float b)
- Point2d operator* (float b, const Point2d &a)
- std::ostream & operator<< (std::ostream &os, const Point2d &point)
- std::istream & operator>> (std::istream &is, Point2d &point)
- bool operator== (const Point2d &a, const Point2d &b)
- bool operator!= (const Point2d &a, const Point2d &b)

9.30.1 Function Documentation

```
9.30.1.1 operator"!=()
bool operator!= (
              const Point2d & a,
              const Point2d & b )
Definition at line 50 of file point2d.cpp.
00051
       return !operator==(a, b);
00052 }
9.30.1.2 operator*() [1/2]
Point2d operator* (
             const Point2d & a,
             float b )
Definition at line 29 of file point2d.cpp.
       return {a.getX() * b, a.getY() * b};
00031 };
9.30.1.3 operator*() [2/2]
Point2d operator* (
            float b,
              const Point2d & a )
Definition at line 33 of file point2d.cpp.
00033 { return operator*(a, b); }
9.30.1.4 operator+()
Point2d operator+ (
             const Point2d & a,
              const Point2d & b )
Definition at line 16 of file point2d.cpp.
                                                           {
00017
       return {a.m_x + b.m_x, a.m_y + b.m_y};
00018 }
9.30.1.5 operator+=()
Point2d & operator+= (
             Point2d & a,
              const Point2d & b )
Definition at line 20 of file point2d.cpp.
```

00020 00021

00022

00023 }

a = a + b;

return a;

9.30.1.6 operator-()

```
Point2d operator- (
            const Point2d & a,
              const Point2d & b )
Definition at line 25 of file point2d.cpp.
00025
00026
                                                           {
       return operator+(a, -1 * b);
00020
9.30.1.7 operator<<()
std::ostream & operator<< (</pre>
             std::ostream & os,
              const Point2d & point )
Definition at line 35 of file point2d.cpp.
       os « "Point2d(" « point.m_x « ", " « point.m_y « ")";
00036
00037
       return os;
00038 }
9.30.1.8 operator==()
bool operator== (
             const Point2d & a,
              const Point2d & b )
Definition at line 46 of file point2d.cpp.
00046
00047
       return a.getX() == b.getX() && a.getY() == b.getY();
00048 }
9.30.1.9 operator>>()
std::istream \& operator>> (
            std::istream & is,
             Point2d & point )
Definition at line 40 of file point2d.cpp.
00041
       is » point.m_x;
00042
       is » point.m_y;
00043
       return is;
00044 }
```

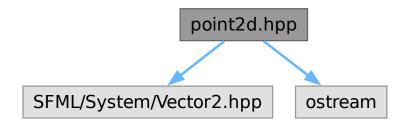
9.31 point2d.cpp 155

9.31 point2d.cpp

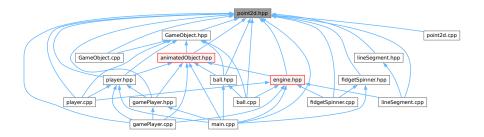
Go to the documentation of this file. 00001 #include "point2d.hpp" 00002 #include "SFML/System/Vector2.hpp" 00003 #include "log.hpp" 00004 #include <cmath> 00005 #include <istream> 00006 #include <iterator> 00007 #include <optional> 00008 #include <ostream> 00009 00010 Point2d::Point2d(cordinate_t x, cordinate_t y) : m_x(x), m_y(y) {}; 00011 Point2d::Point2d(const sf::Vector2f &vector) : m_x{vector.x}, m_y{vector.y} {}; 00012 Point2d::Point2d(const sf::Vector2i &vector) 00013 : m_x{static_cast<cordinate_t>(vector.x)}, 00014 m_y{static_cast<cordinate_t>(vector.y)} {}; 00015 00016 Point2d operator+(const Point2d &a, const Point2d &b) { 00017 return {a.m_x + b.m_x, a.m_y + b.m_y}; 00018 } 00019 00020 Point2d & operator += (Point2d &a, const Point2d &b) { 00021 a = a + b; 00022 return a; 00023 } 00025 Point2d operator-(const Point2d &a, const Point2d &b) { 00026 return operator+(a, -1 * b); 00027 } 00028 00029 Point2d operator*(const Point2d &a, float b) { 00030 return {a.getX() * b, a.getY() * b}; 00031 }; 00032 00033 Point2d operator*(float b, const Point2d &a) { return operator*(a, b); } 00034 00035 std::ostream &operator«(std::ostream &os, const Point2d &point) { 00036 os « "Point2d(" « point.m_x « ", " « point.m_y « ")"; 00037 return os; 00038 } 00039 00040 std::istream &operator (std::istream &is, Point2d &point) { 00041 is » point.m_x; 00042 is » point.m_y; 00042 return is; 00044 } 00045 00046 bool operator==(const Point2d &a, const Point2d &b) { 00047 return a.getX() == b.getX() && a.getY() == b.getY(); 00048 } 00049 00050 bool operator!=(const Point2d &a, const Point2d &b) { 00051 return !operator==(a, b); 00052 } 00053 00054 Point2d::cordinate_t Point2d::getX() const { return m_x; }; 00055 Point2d::cordinate_t Point2d::getY() const { return m_y; }; 00056 00057 void Point2d::setX(cordinate_t x) { m_x = x; }; 00058 void Point2d::setY(cordinate_t y) { m_y = y; }; 00063 void Point2d::swap(Point2d &b) { 00064 std::swap(m_x, b.m_x); std::swap(m_y, b.m_y); 00066 } 00067 00068 sf::Vector2f Point2d::toVector2f() const { 00069 return {static_cast<float>(m_x), static_cast<float>(m_y)}; 00070 } 00072 Point2d::cordinate_t Point2d::length() const { 00074 return length(); 00075 } 00076 00077 Point2d::cordinate_t Point2d::distanceTo(const Point2d &point) const { 00078 return (*this - point).length(); 00079 };

9.32 point2d.hpp File Reference

```
#include "SFML/System/Vector2.hpp"
#include <ostream>
Include dependency graph for point2d.hpp:
```



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



Classes

· class Point2d

Class containing 2D point co-ordinates.

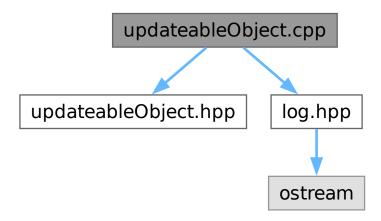
9.33 point2d.hpp

```
00001 #ifndef POINT2D_HPP_
00002 #define POINT2D_HPP_
00003
00004 #include "SFML/System/Vector2.hpp"
00005 #include <ostream>
00009 class Point2d {
00010 public:
00011     using cordinate_t = float;
00012
00013 private:
00017     cordinate_t m_x;
00021     cordinate_t m_y;
00022
00023 public:
00029     Point2d(cordinate_t x = 0, cordinate_t y = 0);
```

```
Point2d(const Point2d &point) = default;
00039
        explicit Point2d(const sf::Vector2f &vector);
00040
00041
        explicit Point2d(const sf::Vector2i &vector);
00042
00043
        friend Point2d operator+(const Point2d &a, const Point2d &b);
00044
        friend Point2d &operator+=(Point2d &a, const Point2d &b);
00045
00046
        friend Point2d operator-(const Point2d &a, const Point2d &b);
00047
00048
        friend bool operator==(const Point2d &a, const Point2d &b);
00049
        friend bool operator!=(const Point2d &a, const Point2d &b);
00050
00051
        friend Point2d operator*(const Point2d &a, float b);
00052
        friend Point2d operator*(float b, const Point2d &a);
00053
00054
        friend std::ostream &operator (std::ostream &os, const Point 2d &point);
00055
00056
        friend std::istream &operator»(std::istream &is, Point2d &point);
00057
00061
        cordinate_t getX() const;
00065
        cordinate_t getY() const;
       void setX(cordinate_t x);
00069
00073
       void setY(cordinate_t y);
00074
00075
       void swap(Point2d &b);
00079
       sf::Vector2f toVector2f() const;
00083
       cordinate_t length() const;
00088
       cordinate_t distanceTo(const Point2d &point) const;
00089 };
00090
00091 #endif // POINT2D_HPP_
```

9.34 updateableObject.cpp File Reference

```
#include "updateableObject.hpp"
#include "log.hpp"
Include dependency graph for updateableObject.cpp:
```



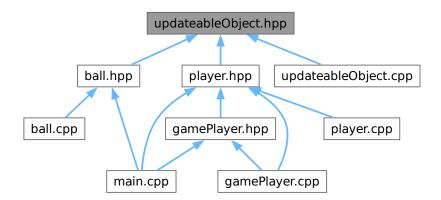
9.35 updateableObject.cpp

```
Go to the documentation of this file.
```

```
00001 #include "updateableObject.hpp"
00002 #include "log.hpp"
00003
00004 // void UpdateableObject::update() { LOGWARN « "not implemented\n"; };
```

9.36 updateableObject.hpp File Reference

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



Classes

· class UpdateableObject

Updateable object class.

9.37 updateableObject.hpp

Go to the documentation of this file.

```
00001 #ifndef UPDATEABLEOBJECT_HPP_
00002 #define UPDATEABLEOBJECT_HPP_
00006 class UpdateableObject {
00007 public:
00011    virtual void update() = 0;
00012 };
00013
00014 #endif // UPDATEABLEOBJECT_HPP_
```

9.38 test/main.cpp File Reference

```
#include "SFML/Graphics/CircleShape.hpp"
#include "SFML/Graphics/RectangleShape.hpp"
#include "SFML/Graphics/RectangleShape.hpp"
#include "SFML/Graphics/Shape.hpp"
#include "SFML/Window/Event.hpp"
#include "engine.hpp"
#include <algorithm>
#include <iostream>
#include <vector>
Include dependency graph for test/main.cpp:
```

animatedObject.hpp log.hpp SFML/Graphics/Drawable.hpp functional set

point2d.hpp vector

algorithm

Namespaces

· namespace G

Functions

- void customLoopFunction ()
- void addSomeRenderables ()
- void setUpCustomEvents ()
- int main ()

Variables

- std::vector< Engine::Shape * > G::drwables {}
- bool G::moveVertically {false}

9.38.1 Function Documentation

9.38.1.1 addSomeRenderables()

```
void addSomeRenderables ( )
```

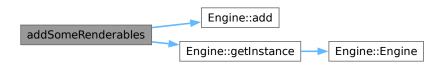
```
Definition at line 39 of file test/main.cpp.
```

```
00040
        Engine &eng = Engine::getInstance();
00041
        const sf::Color pink{0xff, 0x69, 0xb4};
00042
00043
       for (int i{}; i < 8; ++i) {</pre>
00044
         sf::CircleShape *circle
00045
             new sf::CircleShape{10 + static_cast<float>(i) * 4};
00046
         circle->setPosition(50 + i * 100, 100 - static_cast<float>(i) * 4);
00047
         circle->setFillColor(pink);
00048
         eng.add(circle);
         G::drwables.push_back(circle);
00049
00050
00051
00052
        sf::RectangleShape *rectangle{new sf::RectangleShape{{8 * 100 + 50, 4}}};
00053
        rectangle->setPosition(\{50 + 5, 100 + 10\});
00054
        rectangle->setFillColor(pink);
00055
        eng.add(rectangle);
00056
        G::drwables.push_back(rectangle);
00057 }
```

References Engine::add(), G::drwables, eng, and Engine::getInstance().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.38.1.2 customLoopFunction()

void customLoopFunction ()

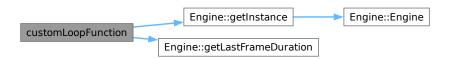
Definition at line 16 of file test/main.cpp.

```
00016
00017
        static float speed = 500;
00018
        static float distanceSum = 0;
00019
00020
        auto deltaTime{Engine::getInstance().getLastFrameDuration().asSeconds()};
00021
        float distance = speed * deltaTime;
00022
00023
       distanceSum += std::abs(distance);
00024
00025
        for (Engine::Shape *i : G::drwables) {
00026
         if (G::moveVertically)
00027
           i->move(distance, 0);
          else (
00028
00029
            i->move(0, distance);
00030
00031
00032
00033
        if (distanceSum >= 500) {
00034
         speed \star = -1;
          distanceSum = 0;
00035
00036
```

 $References\ G:: drwables,\ Engine:: getInstance(),\ Engine:: getLastFrameDuration(),\ and\ G:: moveVertically.$

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.38.1.3 main()

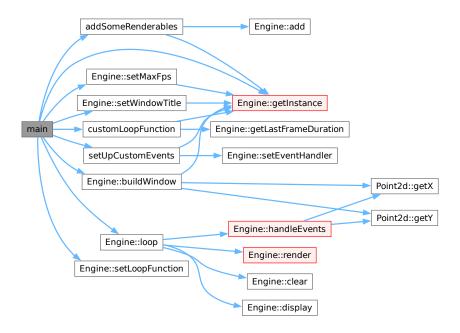
```
int main ( )
```

Definition at line 70 of file test/main.cpp.

```
Engine &eng = Engine::getInstance()
00072
                          .setWindowTitle("dev")
00073
                          .setMaxFps(60)
00074
                          .setLoopFunction(customLoopFunction)
00075
                          .buildWindow();
00076
00077
       setUpCustomEvents();
00078
       addSomeRenderables();
00079
       eng.loop();
00080 }
```

References addSomeRenderables(), Engine::buildWindow(), customLoopFunction(), eng, Engine::getInstance(), Engine::loop(), Engine::setLoopFunction(), Engine::setMaxFps(), setUpCustomEvents(), and Engine::setWindowTitle().

Here is the call graph for this function:



9.38.1.4 setUpCustomEvents()

```
void setUpCustomEvents ( )
```

Definition at line 59 of file test/main.cpp.

```
00059

00060

Engine::getInstance().setEventHandler(
00061

sf::Event::MouseButtonPressed, [] (const sf::Event &ev) {
00062

std::cout « "Custom event handler Mouse button press "

00063

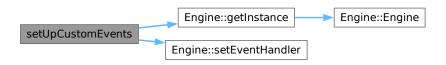
« ev.mouseButton.button « '\t' « ev.mouseButton.x « '\t'
00064

out of the control of th
```

References Engine::getInstance(), G::moveVertically, and Engine::setEventHandler().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.39 test/main.cpp

```
00001 #include "SFML/Graphics/CircleShape.hpp" 00002 #include "SFML/Graphics/Color.hpp"
00003 #include "SFML/Graphics/RectangleShape.hpp"
00004 #include "SFML/Graphics/Shape.hpp"
00005 #include "SFML/Window/Event.hpp"
00006 #include "engine.hpp"
00007 #include <algorithm>
00008 #include <iostream>
00009 #include <vector>
00010
00011 namespace G {
00012 std::vector<Engine::Shape *> drwables{};
00013 bool moveVertically{false};
00014 }; // namespace G
00015
00016 void customLoopFunction() {
        static float speed = 500;
00018
        static float distanceSum = 0;
00019
00020
        auto deltaTime{Engine::getInstance().getLastFrameDuration().asSeconds());
00021
00022
         float distance = speed * deltaTime;
         distanceSum += std::abs(distance);
00023
00024
00025
         for (Engine::Shape *i : G::drwables) {
00026
          if (G::moveVertically)
             i->move(distance, 0);
00027
00028
           else {
00029
             i->move(0, distance);
00030
00031
00032
00033
         if (distanceSum >= 500) {
00034
           speed \star = -1;
00035
           distanceSum = 0;
00036
```

```
00037 }
00038
00039 void addSomeRenderables() {
00040 Engine &eng = Engine::getInstance();
00041
       const sf::Color pink{0xff, 0x69, 0xb4};
00042
      for (int i{}; i < 8; ++i) {</pre>
       sf::CircleShape *circle =
00044
00045
           new sf::CircleShape{10 + static_cast<float>(i) * 4};
         circle->setPosition(50 + i * 100, 100 - static_cast<float>(i) * 4);
00046
00047
         circle->setFillColor(pink);
00048
         eng.add(circle);
00049
         G::drwables.push_back(circle);
00050 };
00051
00052 sf::RectangleShape *rectangle{new } sf::RectangleShape{8 * 100 + 50, 4}};
       rectangle->setPosition({50 + 5, 100 + 10});
00053
       rectangle->setFillColor(pink);
00054
00055
       eng.add(rectangle);
00056 G::drwables.push_back(rectangle);
00057 }
00058
00059 void setUpCustomEvents() {
00060 Engine::getInstance().setEventHandler(
00061
           sf::Event::MouseButtonPressed, [] (const sf::Event &ev) {
            std::cout « "Custom event handler Mouse button press "
00063
                      « ev.mouseButton.button « '\t' « ev.mouseButton.x « '\t'
            00064
00065
00066
              G::moveVertically = !G::moveVertically;
00067
00068 }
00069
00070 int main() {
00071 Engine &eng = Engine::getInstance()
                        .setWindowTitle("dev")
00072
00073
                         .setMaxFps(60)
                         .setLoopFunction(customLoopFunction)
00075
                         .buildWindow();
00076
00077
       setUpCustomEvents();
00078
       addSomeRenderables();
00079
       eng.loop();
00080 }
```

9.40 test2/main.cpp File Reference

```
#include "SFML/Graphics/Color.hpp"
#include "SFML/Graphics/Rect.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "SFML/Graphics/Texture.hpp"
#include "SFML/System/Vector2.hpp"
#include "SFML/Window/Event.hpp"
#include "SFML/Window/Keyboard.hpp"
#include "animatedObject.hpp"
#include "animatedSpriteSheet.hpp"
#include "ball.hpp"
#include "bush.hpp"
#include "engine.hpp"
#include "fidgetSpinner.hpp"
#include "gamePlayer.hpp"
#include "log.hpp"
#include "player.hpp"
#include "point2d.hpp"
#include <cstdlib>
#include <iostream>
#include <list>
#include <random>
```

#include <utility>

Include dependency graph for test2/main.cpp:



Functions

- Ball * throwBall (Point2d pos, Point2d vector)
- sf::Color getRandomColor ()
- void throwBallPlayer ()
- void initialziePlayer ()
- void initializeBush ()
- void handlePlayerMovement ()
- void handleBalls ()
- void customLoop ()
- void spinTheFidget ()
- void keyPressedEventHandler (const sf::Event &ev)
- void keyReleasedEventHandler (const sf::Event &ev)
- void initializeFidgetSpinner ()
- int main ()

Variables

- GamePlayer player {}
- std::array< Bush, 4 > bush {}
- Engine & eng = Engine::getInstance()
- FidgetSpinner fidgetSpinner {"resource/fidgetSpinner"}
- std::list< Ball * > balls

9.40.1 Function Documentation

9.40.1.1 customLoop()

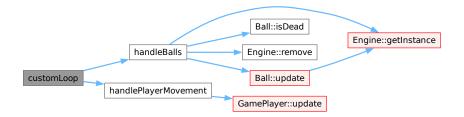
```
void customLoop ( )
```

Definition at line 94 of file test2/main.cpp.

References handleBalls(), and handlePlayerMovement().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.2 getRandomColor()

```
sf::Color getRandomColor ( )
```

Definition at line 41 of file test2/main.cpp.

```
00041 {
00042 return {static_cast<sf::Uint8>(rand() % 256),
00043 static_cast<sf::Uint8>(rand() % 256),
00044 static_cast<sf::Uint8>(rand() % 256)};
00045 }
```

Referenced by throwBallPlayer().

Here is the caller graph for this function:



9.40.1.3 handleBalls()

```
void handleBalls ( )
```

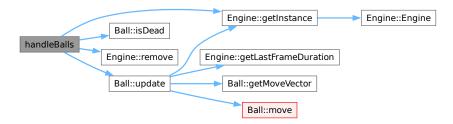
Definition at line 79 of file test2/main.cpp.

```
08000
00081
         for (auto it{balls.begin()}; it != balls.end();) {
00082
          auto it2{it};
00083
           ++it;
00084
           Ball *b = *it2;
b->update();
if (b->isDead()) {
00085
00086
00087
             Engine::getInstance().remove(b);
88000
             balls.remove(b);
00089
             delete b;
00090
00091
        }
00092 }
```

References balls, Engine::getInstance(), Ball::isDead(), Engine::remove(), and Ball::update().

Referenced by customLoop().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.4 handlePlayerMovement()

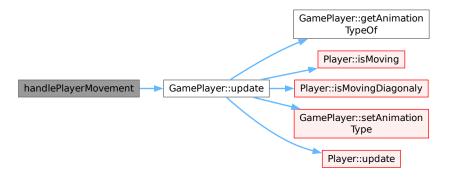
```
void handlePlayerMovement ( )
```

Definition at line 77 of file test2/main.cpp. 00077 { player.update(); }

References player, and GamePlayer::update().

Referenced by customLoop().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.5 initializeBush()

```
void initializeBush ( )
```

Definition at line 66 of file test2/main.cpp.

References Engine::add(), bush, eng, and LOGINFO.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.6 initializeFidgetSpinner()

```
void initializeFidgetSpinner ( )
```

Definition at line 163 of file test2/main.cpp.

```
00163
00164 fidgetSpinner.setPosition({600, 600});
00165 eng.add(&fidgetSpinner);
00166 }
```

References Engine::add(), eng, fidgetSpinner, and AnimatedObject::setPosition().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.7 initialziePlayer()

```
void initialziePlayer ( )
```

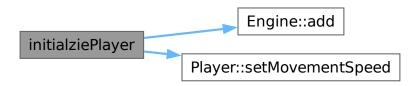
Definition at line 59 of file test2/main.cpp.

```
00059 {
00060 LOGINFON;
00061 player.setMovementSpeed(50);
00062 player.setColor(sf::Color::Yellow);
00063 eng.add(&player);
00064 }
```

References Engine::add(), eng, LOGINFON, player, and Player::setMovementSpeed().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.8 keyPressedEventHandler()

00123 00124

00125

break;

case sf::Keyboard::D:

```
void keyPressedEventHandler (
                const sf::Event & ev )
Definition at line 114 of file test2/main.cpp.
        switch (ev.key.code) {
00115
        case sf::Keyboard::W:
  player.setIsMoving(Player::MoveDirection::north);
00116
00117
00118
          break;
00119
        case sf::Keyboard::A:
        player.setIsMoving(Player::MoveDirection::west);
break;
00120
00121
00122
        case sf::Keyboard::S:
```

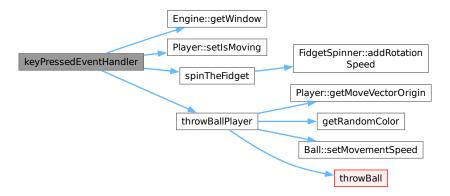
player.setIsMoving(Player::MoveDirection::south);

```
player.setIsMoving(Player::MoveDirection::east);
00127
00128
00129
       case sf::Keyboard::F:
00130
        throwBallPlayer();
00131
          break:
00132
       case sf::Keyboard::Space:
        spinTheFidget();
break;
00133
00134
00135
       case sf::Keyboard::Escape:
        eng.getWindow().close();
break;
00136
00137
00138
00139
00140
00141
00142 }
```

References Player::east, eng, Engine::getWindow(), Player::north, player, Player::setIsMoving(), Player::south, spinTheFidget(), throwBallPlayer(), and Player::west.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.9 keyReleasedEventHandler()

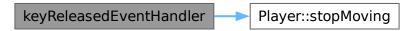
Definition at line 144 of file test2/main.cpp.

```
00144
                                                      {
00145
       switch (ev.key.code) {
00146
       case sf::Keyboard::W:
       player.stopMoving(Player::MoveDirection::north);
00147
00148
         break;
00149
       case sf::Keyboard::A:
       player.stopMoving(Player::MoveDirection::west);
00150
00151
00152
       case sf::Keyboard::S:
       player.stopMoving(Player::MoveDirection::south);
00153
00154
         break;
00155
       case sf::Keyboard::D:
       player.stopMoving(Player::MoveDirection::east);
break;
00156
00157
00158
       default:
00159
         break;
       }
00160
00161 }
```

References Player::east, Player::north, player, Player::south, Player::stopMoving(), and Player::west.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.10 main()

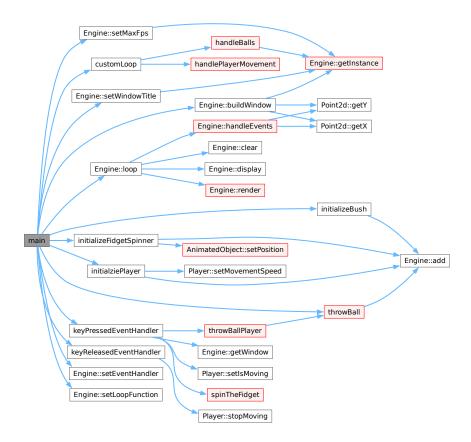
```
int main ( )
```

Definition at line 168 of file test2/main.cpp.

```
00168
        LOGTRACEN;
00170
        eng.setWindowTitle("dev").setMaxFps(75).buildWindow();
00171
        eng.setEventHandler(sf::Event::KeyPressed, keyPressedEventHandler);
00172
        \verb|eng.setEventHandler| (\verb|sf::Event::KeyReleased|, keyReleasedEventHandler|); \\
00173
        eng.setLoopFunction(customLoop);
00174
00175
        initialziePlayer();
00176
        initializeBush();
00177
        initializeFidgetSpinner();
00178
        throwBall({100, 0}, {1, 1});
00179
00180
        eng.loop();
00181 }
```

References Engine::buildWindow(), customLoop(), eng, initializeBush(), initializeFidgetSpinner(), initializePlayer(), keyPressedEventHandler(), keyReleasedEventHandler(), LOGTRACEN, Engine::loop(), Engine::setEventHandler(), Engine::setLoopFunction(), Engine::setMaxFps(), Engine::setWindowTitle(), and throwBall().

Here is the call graph for this function:



9.40.1.11 spinTheFidget()

```
void spinTheFidget ( )
```

Definition at line 99 of file test2/main.cpp.

```
00100
00101
       Point2d pla{Point2d(player.sf::Sprite::getPosition()) +
00102
                    Point2d{player.sf::Sprite::getTextureRect().getSize()} * 0.5};
       Point2d fig{Point2d{fidgetSpinner.sf::Sprite::getPosition()} +
00103
00104
                    Point2d{fidgetSpinner.getTextureRect().getSize()} * 0.5};
00105
00106
       float dist(pla.distanceTo(fig));
00107
00108
       LOGINFO « dist « '\n';
       if (dist < 300) {</pre>
00109
         fidgetSpinner.addRotationSpeed(8);
00110
00111
00112 }
```

References FidgetSpinner::addRotationSpeed(), fidgetSpinner, LOGINFO, and player.

Referenced by keyPressedEventHandler().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.12 throwBall()

```
Ball * throwBall (

Point2d pos,

Point2d vector )

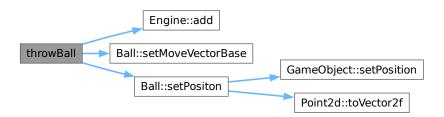
Definition at line 32 of file test2/main.cpp.

00032
```

References Engine::add(), balls, eng, Ball::setMoveVectorBase(), and Ball::setPositon().

Referenced by main(), and throwBallPlayer().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.1.13 throwBallPlayer()

```
void throwBallPlayer ( )
```

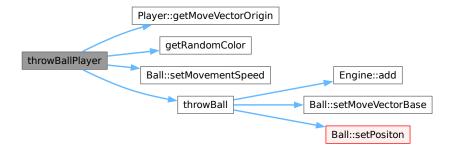
Definition at line 47 of file test2/main.cpp.

```
00048
        Ball *bal{throwBall(
00049
            static_cast<Point2d>(
00050
                player.GameObject::getPosition() +
00051
                static_cast<Point2d>(player.sf::Sprite::getTextureRect().getSize()) *
00052
                    0.5),
00053
            player.getMoveVectorOrigin())};
00054
00055
        bal->setMovementSpeed(80);
00056
        bal->setFillColor(getRandomColor());
00057 }
```

References Player::getMoveVectorOrigin(), getRandomColor(), player, Ball::setMovementSpeed(), and throwBall().

Referenced by keyPressedEventHandler().

Here is the call graph for this function:



Here is the caller graph for this function:



9.40.2 Variable Documentation

9.40.2.1 balls

```
std::list<Ball *> balls
```

Definition at line 30 of file test2/main.cpp.

Referenced by handleBalls(), and throwBall().

9.40.2.2 bush

```
std::array<Bush, 4> bush {}
```

Definition at line 27 of file test2/main.cpp. 00027 {};

Referenced by initializeBush().

9.40.2.3 eng

```
Engine& eng = Engine::getInstance()
```

Definition at line 28 of file test2/main.cpp.

Referenced by addSomeRenderables(), initializeBush(), initializeFidgetSpinner(), initialziePlayer(), keyPressedEventHandler(), main(), and throwBall().

9.40.2.4 fidgetSpinner

```
FidgetSpinner fidgetSpinner {"resource/fidgetSpinner"}
```

Definition at line 29 of file test2/main.cpp. 00029 {"resource/fidgetSpinner"};

Referenced by initializeFidgetSpinner(), and spinTheFidget().

9.40.2.5 player

```
GamePlayer player {}
```

Definition at line 26 of file test2/main.cpp. 00026 {};

Referenced by handlePlayerMovement(), initialziePlayer(), keyPressedEventHandler(), keyReleasedEventHandler(), spinTheFidget(), and throwBallPlayer().

9.41 test2/main.cpp

```
Go to the documentation of this file.
00001 #include "SFML/Graphics/Color.hpp"
00002 #include "SFML/Graphics/Rect.hpp"
00003 #include "SFML/Graphics/Sprite.hpp"
00004 #include "SFML/Graphics/Texture.hpp"
00005 #include "SFML/System/Vector2.hpp
00006 #include "SFML/Window/Event.hpp"
00007 #include "SFML/Window/Keyboard.hpp"
00008 #include "animatedObject.hpp"
00009 #include "animatedSpriteSheet.hpp'
00010 #include "ball.hpp"
00011 #include "bush.hpp"
00011 #include "busn.npp"
00012 #include "engine.hpp"
00013 #include "fidgetSpinner.hpp"
00014 #include "gamePlayer.hpp"
00015 #include "log.hpp"
00016 #include "player.hpp"
00017 #include "point2d.hpp"
00018 #include <cstdlib>
00019 #include <iostream>
00020 #include <list>
00021 #include <random>
00022 #include <utility>
00024 using sf::IntRect;
00025
00026 GamePlayer player{};
00027 std::array<Bush, 4> bush{};
00028 Engine &eng = Engine::getInstance();
00029 FidgetSpinner fidgetSpinner("resource/fidgetSpinner");
00030 std::list<Ball *> balls;
00031
00032 Ball *throwBall(Point2d pos, Point2d vector) {
00033
        Ball *bal = new Ball();
        bal->setPositon(pos);
00034
00035
         bal->setMoveVectorBase(vector);
00036
         balls.push_back(bal);
00037
         eng.add(bal);
00038
        return bal;
00039 }
00040
00041 sf::Color getRandomColor() {
00043
                 static_cast<sf::Uint8>(rand() % 256),
00044
                 static_cast<sf::Uint8>(rand() % 256)};
00045 }
00046
00047 void throwBallPlayer() {
00048 Ball *bal{throwBall(
00049
          static_cast<Point2d>(
00050
               player.GameObject::getPosition() +
00051
                  static_cast<Point2d>(player.sf::Sprite::getTextureRect().getSize()) *
00052
                      0.5),
             player.getMoveVectorOrigin())};
00053
00054
00055
        bal->setMovementSpeed(80);
00056
        bal->setFillColor(getRandomColor());
00057 }
00058
00059 void initialziePlayer() {
00060 LOGINFON;
00061
         player.setMovementSpeed(50);
00062
         player.setColor(sf::Color::Yellow);
00063
         eng.add(&player);
00064 }
00065
00066 void initializeBush() {
00067
      for (ssize_t i{}; i < bush.max_size(); ++i) {
   LOGINFO « i « '\n';</pre>
00068
00069
          eng.add(&(bush[i]));
00070
00071
        bush[0].setPosition({300, 150});
00072
         bush[1].setPosition({700, 550});
         bush[2].setPosition({100, 450});
00074
         bush[3].setPosition({800, 250});
00075 }
00076
00077 void handlePlayerMovement() { player.update(); }
00078
00079 void handleBalls() {
08000
00081
         for (auto it{balls.begin()}; it != balls.end();) {
00082
           auto it2{it};
```

9.41 test2/main.cpp 177

```
00083
          ++it;
00084
          Ball *b = *it2;
00085
          b->update();
00086
          if (b->isDead()) {
00087
            Engine::getInstance().remove(b);
00088
            balls.remove(b);
00089
            delete b;
00090
00091
       }
00092 }
00093
00094 void customLoop() {
00095
        handlePlayerMovement();
00096
        handleBalls();
00097 }
00098
00099 void spinTheFidget() {
00100
00101
        Point2d pla{Point2d(player.sf::Sprite::getPosition()) +
00102
                    Point2d{player.sf::Sprite::getTextureRect().getSize()} * 0.5};
00103
        Point2d fig{Point2d{fidgetSpinner.sf::Sprite::getPosition()} +
00104
                    Point2d{fidgetSpinner.getTextureRect().getSize()} * 0.5};
00105
00106
       float dist(pla.distanceTo(fig));
00107
       LOGINFO « dist « '\n';
00108
00109
        if (dist < 300) {</pre>
00110
         fidgetSpinner.addRotationSpeed(8);
00111
00112 }
00113
00114 void keyPressedEventHandler(const sf::Event &ev) {
00115
       switch (ev.key.code) {
00116
        case sf::Keyboard::W:
00117
        player.setIsMoving(Player::MoveDirection::north);
00118
         break;
00119
        case sf::Keyboard::A:
        player.setIsMoving(Player::MoveDirection::west);
00121
          break;
00122
        case sf::Keyboard::S:
        player.setIsMoving(Player::MoveDirection::south);
00123
00124
         break:
00125
        case sf::Keyboard::D:
        player.setIsMoving(Player::MoveDirection::east);
break;
00126
00127
00128
00129
       case sf::Keyboard::F:
        throwBallPlayer();
00130
00131
         break:
00132
        case sf::Keyboard::Space:
        spinTheFidget();
break;
00133
00134
00135
       case sf::Keyboard::Escape:
00136
        eng.getWindow().close();
break;
00137
00138
00139
       default:
00140
         break;
00141
00142 }
00143
00144 void keyReleasedEventHandler(const sf::Event &ev) {
       switch (ev.key.code) {
00146
       case sf::Keyboard::W:
        player.stopMoving(Player::MoveDirection::north);
00147
00148
         break;
00149
        case sf::Keyboard::A:
        player.stopMoving(Player::MoveDirection::west);
00150
00151
         break:
00152
        case sf::Keyboard::S:
        player.stopMoving(Player::MoveDirection::south);
00153
00154
         break;
00155
        case sf::Keyboard::D:
00156
        player.stopMoving(Player::MoveDirection::east);
00157
          break:
00158
        default:
00159
         break;
00160
00161 }
00162
00163 void initializeFidgetSpinner() {
00164
       fidgetSpinner.setPosition({600, 600});
00165
        eng.add(&fidgetSpinner);
00166 }
00167
00168 int main() {
00169
       LOGTRACEN;
```

```
eng.setWindowTitle("dev").setMaxFps(75).buildWindow();
00171
        eng.setEventHandler(sf::Event::KeyPressed, keyPressedEventHandler);
00172
        eng.setEventHandler(sf::Event::KeyReleased, keyReleasedEventHandler);
00173
       eng.setLoopFunction(customLoop);
00174
00175
       initialziePlayer();
00176
       initializeBush();
00177
       initializeFidgetSpinner();
00178
       throwBall({100, 0}, {1, 1});
00179
       eng.loop();
00180
00181 }
```

9.42 test3/main.cpp File Reference

```
#include "SFML/Graphics/Color.hpp"
#include "animatedSpriteSheet.hpp"
#include "engine.hpp"
#include <iostream>
```

Include dependency graph for test3/main.cpp:



Namespaces

· namespace G

Functions

• int main ()

Variables

- std::string G::basePath = "resources/"
- AnimatedSpriteSheet animation (G::basePath+"animation")

9.42.1 Function Documentation

9.42.1.1 main()

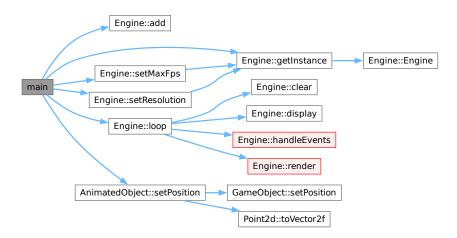
```
int main ( )
```

Definition at line 12 of file test3/main.cpp.

9.43 test3/main.cpp 179

References Engine::add(), animation, Engine::getInstance(), Engine::loop(), Engine::setMaxFps(), AnimatedObject::setPosition(), and Engine::setResolution().

Here is the call graph for this function:



9.42.2 Variable Documentation

9.42.2.1 animation

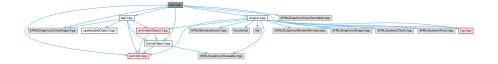
Referenced by main(), and GamePlayer::update().

9.43 test3/main.cpp

```
00001 #include "SFML/Graphics/Color.hpp"
00002 #include "animatedSpriteSheet.hpp"
00003 #include "engine.hpp"
00004 #include <iostream>
00005
00006 namespace G {
00007 std::string basePath = "resources/";
00008 }; // namespace G
00009
00010 AnimatedSpriteSheet animation(G::basePath + "animation");
00011
00012 int main() {
00013
        Engine::getInstance().setMaxFps(3).setResolution({1000, 1000}).buildWindow();
00014
         animation.setPosition({300, 300});
00015
00016
        animation.setColor(sf::Color::Cvan);
00017
00018
         Engine::getInstance().add(&animation);
00019
00020
         Engine::getInstance().loop();
00021 }
```

9.44 ball.cpp File Reference

```
#include "ball.hpp"
#include "GameObject.hpp"
#include "SFML/Graphics/CircleShape.hpp"
#include "SFML/Graphics/Transformable.hpp"
#include "engine.hpp"
#include "log.hpp"
#include "point2d.hpp"
Include dependency graph for ball.cpp:
```



9.45 ball.cpp

Go to the documentation of this file.

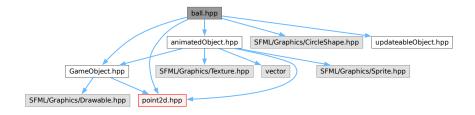
```
00001 #include "ball.hpp"
00002 #include "GameObject.hpp"
00003 #include "SFML/Graphics/CircleShape.hpp"
00004 #include "SFML/Graphics/Transformable.hpp"
00005 #include "engine.hpp"
00006 #include "log.hpp"
00007 #include "point2d.hpp"
80000
00009 Ball::Ball(float radius, std::size_t pointCount)
00010
         : sf::CircleShape(radius, pointCount) {}
00011
00012 void Ball::setMovementSpeed(float movementSpeed) {
00013
        m_movementSpeed = movementSpeed;
00014 }
00015
00016 void Ball::setMoveVectorBase(Point2d moveVectorBase) {
00017
       m_moveVectorBase = moveVectorBase;
00018 }
00019
00020 void Ball::move(Point2d vector) {
00021
       GameObject::move(vector);
00022
       sf::CircleShape::move(vector.toVector2f());
00023 }
00024
00025 void Ball::setPositon(Point2d pos) {
00026 GameObject::setPosition(pos);
00027 sf::Transformable::setPosition(pos.toVector2f());
00028 }
00029
00030 bool Ball::isDead() const { return m dead; }
00031
00032 void Ball::update() {
00034
        m_dead = true;
00035
          return:
00036
00037
        m_timeToLife -= Engine::getInstance().getLastFrameDuration().asSeconds();
00039
        move(getMoveVector() *
00040
              Engine::getInstance().getLastFrameDuration().asSeconds());
00041 }
00042
00043 Point2d Ball::qetMoveVector() { return {m_moveVectorBase * m_movementSpeed}; }
```

9.46 ball.hpp File Reference

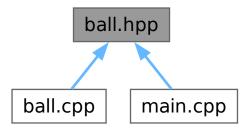
```
#include "GameObject.hpp"
#include "SFML/Graphics/CircleShape.hpp"
```

9.47 ball.hpp 181

```
#include "animatedObject.hpp"
#include "point2d.hpp"
#include "updateableObject.hpp"
Include dependency graph for ball.hpp:
```



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



Classes

• class Ball

9.47 ball.hpp

```
00001 #ifndef BALL_HPP_
00002 #define BALL_HPP_
00003
00004 #include "GameObject.hpp"
00005 #include "SFML/Graphics/CircleShape.hpp"
00006 #include "animatedObject.hpp"
00007 #include "point2d.hpp"
00008 #include "updateableObject.hpp"
00009
00010 class Ball : public GameObject,
                         public sf::CircleShape,
public UpdateableObject {
00011
00012
00013 public:
00014 Ball(float radius = 10, std::size_t pointCount = 30);
00015
00016
         void setMovementSpeed(float movementSpeed);
00017
          void setMoveVectorBase(Point2d moveVectorOrigin);
00018
00019
         virtual void update();
         virtual void move (Point2d vector);
```

```
virtual void setPositon(Point2d pos);
00022
00023
       bool isDead() const;
00024
00025 private:
00026
        Point2d getMoveVector();
00028 private:
00029
       float m_movementSpeed{10};
00030
       Point2d m_moveVectorBase{};
       float m_timeToLife{15};
00031
00032
       bool m_dead{false};
00033 };
00034
00035 #endif // BALL_HPP_
```

9.48 bush.cpp File Reference

```
#include "bush.hpp"
#include "SFML/Graphics/Color.hpp"
#include "SFML/Graphics/Rect.hpp"
#include "engine.hpp"
#include "log.hpp"
#include <system_error>
Include dependency graph for bush.cpp:
```

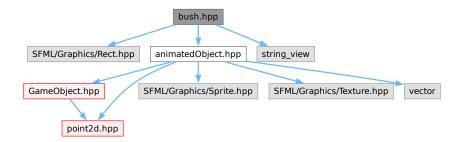


9.49 bush.cpp

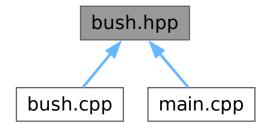
```
00001 #include "bush.hpp"
00002 #include "SFML/Graphics/Color.hpp"
00002 #Include "SFML/Graphics/Rect.hpp"
00003 #include "engine.hpp"
00005 #include "log.hpp"
00006 #include <system_error>
00007
00008 Bush::~Bush(){};
00009
00010 Bush::Bush() {
00011
         LOGINFON;
         loadFromFile(std::string(s_spriteSheetPath), sf::IntRect{0, 0, 300, 150});
00012
00013
         setTexture(*this);
00014
        setTextureRect(getCurrentSpriteRectangle(0));
00016
00017 void Bush::animate() {
00018    m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
00019    if (m_animationTimer < m_animationFrameDuration)</pre>
00020
           return;
00021
00022
         m_animationTimer = 0;
00023 nextSprite();
00024 }
00025
00026 void Bush::nextSprite() {
        this->setTextureRect(getCurrentSpriteRectangle(m_aninmationFrameIndicator));
00028
        ++m_aninmationFrameIndicator;
         m_aninmationFrameIndicator %= 2;
00029
00030 }
00031
00032 sf::IntRect Bush::getCurrentSpriteRectangle(int frameNumber) {
00033
        return sf::IntRect{frameNumber * 150, 0, 150, 150};
00034 }
```

9.50 bush.hpp File Reference

```
#include "SFML/Graphics/Rect.hpp"
#include "animatedObject.hpp"
#include <string_view>
Include dependency graph for bush.hpp:
```



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



Classes

· class Bush

9.51 bush.hpp

```
00001 #ifndef BUSH_HPP_
00002 #define BUSH_HPP_
00003
00004 #include "SFML/Graphics/Rect.hpp"
00005 #include "animatedObject.hpp"
00006 #include <string_view>
00007
00008 class Bush : public AnimatedObject {
00009    static constexpr std::string_view s_spriteSheetPath{"resource/bush/bush.png"};
00010    int m_animationFrameIndicator{};
00011    float m_animationFrameDuration{1};
00012    float m_animationTimer{};
```

```
00013
00014 public:
00015
       Bush();
00016
       ~Bush();
00017
00018
       virtual void animate();
00020 private:
00021 void nextSprite();
00022
       sf::IntRect getCurrentSpriteRectangle(int frameNumber);
00023 };
00024
00025 #endif // BUSH_HPP_
```

9.52 fidgetSpinner.cpp File Reference

```
#include "fidgetSpinner.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "animatedSpriteSheet.hpp"
#include "engine.hpp"
#include "log.hpp"
#include "point2d.hpp"
Include dependency graph for fidgetSpinner.cpp:
```



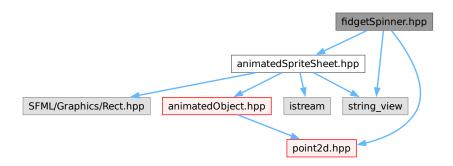
9.53 fidgetSpinner.cpp

```
Go to the documentation of this file.
```

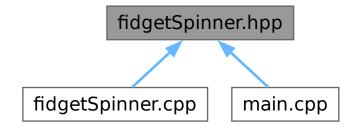
```
00001 #include "fidgetSpinner.hpp"
00002 #include "SFML/Graphics/Sprite.hpp"
00003 #include "animatedSpriteSheet.hpp
00003 #include "engine.hpp"
00005 #include "log.hpp"
00006 #include "point2d.hpp'
00007
00008 FidgetSpinner::FidgetSpinner(std::string_view path)
00009
          : AnimatedSpriteSheet(path) {
00010
        LOGINFON:
00011
        Point2d offset(getSize().x / 2.f + 0.5f, getSize().y / 2.f + 0.5f);
00012
00013
        setOrigin(offset.toVector2f());
00014 };
00015
00016 void FidgetSpinner::animate() {
00017
       00018
00019
        setRotation(m angle);
00020
00021
        if (m_rotationspeed > 0) {
00022
          m_rotationspeed -= m_rotationResistance + m_rotationspeed / 500;
00023
          if (m_rotationspeed < 0)</pre>
00024
            m_rotationspeed = 0;
00025
00026 }
00027
00028 Point2d FidgetSpinner::getCenterPoint() const {
00029
       Point2d point{getCurrentAnimationFrameData().m_position +
00030
                      getCurrentAnimationFrameData().m_size * 0.5};
00031
00032
        return point;
00033 }
00034
00035 void FidgetSpinner::addRotationSpeed(float speed) { m_rotationspeed += speed; }
00036
00037 void FidgetSpinner::setRotationResistance(float speed) {
00038
       m_rotationResistance = speed;
00039 }
```

9.54 fidgetSpinner.hpp File Reference

```
#include "animatedSpriteSheet.hpp"
#include "point2d.hpp"
#include <string_view>
Include dependency graph for fidgetSpinner.hpp:
```



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



Classes

class FidgetSpinner

9.55 fidgetSpinner.hpp

```
00001 #ifndef FIDGETSPINNER_HPP_
00002 #define FIDGETSPINNER_HPP_
00003
00004 #include "animatedSpriteSheet.hpp"
00005 #include "point2d.hpp"
00006 #include <string_view>
00007
00008 class FidgetSpinner : public AnimatedSpriteSheet {
```

```
00010 public:
00011
       FidgetSpinner(std::string_view path);
00012
00013
       virtual void animate() override;
00014
00015
       Point2d getCenterPoint() const;
00016
00017
       void addRotationSpeed(float speed);
00018
00019
       void setRotationResistance(float speed);
00020
00021 private:
00022
        float m_rotationResistance{0.1};
00023
       float m_rotationspeed{0};
00024
        float m_angle{};
00025 };
00026
00027 #endif // FIDGETSPINNER_HPP_
```

9.56 gamePlayer.cpp File Reference

```
#include "gamePlayer.hpp"
#include "SFML/Graphics/Rect.hpp"
#include "SFML/Graphics/Sprite.hpp"
#include "animatedObject.hpp"
#include "engine.hpp"
#include "log.hpp"
#include "player.hpp"
#include "point2d.hpp"
#include <cmath>
#include <sys/types.h>
Include dependency graph for gamePlayer.cpp:
```



9.57 gamePlayer.cpp

```
00002 #include "gamePlayer.hpp"
00002 #Include "gammerlayer.hpp"
00003 #include "SFML/Graphics/Rect.hpp"
00004 #include "SFML/Graphics/Sprite.hpp"
00005 #include "animatedObject.hpp"
00006 #include "engine.hpp"
00007 #include "log.hpp"
00008 #include "player.hpp"
00009 #include "point2d.hpp"
00010 #include <cmath>
00011 #include <sys/types.h>
00012
00013 // #define NOTRACE
00014
00015 GamePlayer::GamePlayer() {
         LOGINFON;
00016
         loadFromFile(
00017
00018
               std::string(s_spriteSheetPath),
00019
               sf::IntRect{0, 0, 4 * 300, static_cast<int>(AnimationType::count) * 300});
         setTexture(*this);
00021
         setTextureRect(getCurrentAnimationFrameSpriteRectangle());
00022 }
00023
```

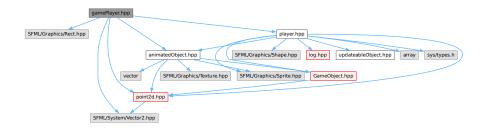
9.57 gamePlayer.cpp 187

```
00024 sf::IntRect GamePlayer::getCurrentAnimationFrameSpriteRectangle() const {
00025
        sf::IntRect result{m_aninmationFrameIndicator * 300,
        getRowOfAnimation() * 300, 300, 300};
// LOGINFO « result.top « '\t' « result.left « '\n';
00026
00027
00028
        return result;
00029 }
00030
00031 void GamePlayer::animate() {
00032
       m_animationTimer += Engine::getInstance().getLastFrameDuration().asSeconds();
00033
        // wait for next frame change
        if (m_animationTimer < m_animationFrameDuration)</pre>
00034
00035
         return;
00036
        // reset timer
00037
        m_animationTimer = 0;
00038
        // do frame changing
00039
       nextAnimationFrame();
00040 }
00041
00042 void GamePlayer::update() {
00043
       Player::update();
00044
00045
        // Handle animation update
00046
        AnimationType animation = AnimationType::standing;
00047
00048
        if (isMoving()) {
00049
         // if is moving diagonally
00050
          if (isMovingDiagonaly()) {
00051
            animation = AnimationType::moveSpecial;
          } else {
00052
            \ensuremath{//} check if is moving in any single direction
00053
00054
            for (int i{}; i < static_cast<int>(MoveDirection::count); ++i) {
00055
              MoveDirection moveDir{static_cast<MoveDirection>(i)};
00056
              if (isMoving(moveDir)) {
00057
                animation = getAnimationTypeOf(moveDir);
00058
                break;
00059
00060
            }
00061
         }
00062
00063
00064
        setAnimationType(animation);
00065 }
00066
00067 int GamePlayer::getFrameCountOfAnimation(AnimationType type) const {
00068
       switch (type) {
00069
        case AnimationType::standing:
00070
         return 4:
00071
        case AnimationType::moveNorth:
00072
         return 4:
00073
        case AnimationType::moveEast:
00074
         return 4;
00075
        case AnimationType::moveSouth:
00076
         return 4;
00077
       case AnimationType::moveWest:
00078
         return 4:
00079
       case AnimationType::moveSpecial:
08000
         return 4;
00081
        default:
00082
        LOGWARN « "Animation type not handled!\n";
00083
          return 0;
00084
         break:
00085
       }
00086 }
00087
00088 void GamePlayer::setAnimationType (GamePlayer::AnimationType type) {
00089
       // Do no reset current animation if new is same.
00090
        if (type == m_animationType)
00091
         return:
00092
        m_animationType = type;
        m_aninmationFrameIndicator = 0;
00093
00094
        m_animationTimer = 0;
00095
       nextAnimationFrame();
00096 };
00097
00098 GamePlayer::AnimationType
00099 GamePlayer::getAnimationTypeOf(MoveDirection moveDirection) const {
00100
       switch (moveDirection) {
00101
        case Player::MoveDirection::north:
00102
         return AnimationType::moveNorth;
00103
        case Player::MoveDirection::east:
         return AnimationType::moveEast;
00104
00105
        case Player::MoveDirection::south:
00106
         return AnimationType::moveSouth;
00107
        case Player::MoveDirection::west:
00108
         return AnimationType::moveWest;
00109
        default:
00110
          LOGWARN « "UNHANDLED\n";
```

```
00111
          break;
00112
00113
        return AnimationType::standing;
00114 }
00115
00116 void GamePlayer::nextAnimationFrame() {
00117
        updateTextureRect();
00118
         ++m_aninmationFrameIndicator;
        m_aninmationFrameIndicator %= getFrameCountOfAnimation(m_animationType);
// LOGINFO « "Frame: " « m_aninmationFrameIndicator
// « " Type: " « static_cast<int>(m_animationType) « '\n';
00119
00120
00121
00122 };
00123
00124 void GamePlayer::updateTextureRect() {
00125
        this->setTextureRect(getCurrentAnimationFrameSpriteRectangle());
00126 }
00127
00128 int GamePlayer::getRowOfAnimation() const {
        switch (m_animationType) {
00130
        case AnimationType::standing:
00131
          return 0;
00132
        case AnimationType::moveNorth:
00133
          return 1;
        case AnimationType::moveEast:
00134
00135
          return 2;
00136
        case AnimationType::moveSouth:
00137
00138
        case AnimationType::moveWest:
00139
          return 4;
        case AnimationType::moveSpecial:
00140
00141
          return 5;
00142
        default:
        LOGWARN « "not handled\n";
00143
00144
          return 0;
00145
        };
00146 };
```

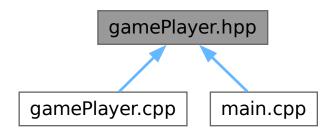
9.58 gamePlayer.hpp File Reference

```
#include "SFML/Graphics/Rect.hpp"
#include "SFML/System/Vector2.hpp"
#include "animatedObject.hpp"
#include "player.hpp"
#include "point2d.hpp"
Include dependency graph for gamePlayer.hpp:
```



9.59 gamePlayer.hpp 189

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



Classes

class GamePlayer

9.59 gamePlayer.hpp

```
00001 #ifndef GAMEPLAYER_H_
00002 #define GAMEPLAYER_H_
00003
00004 #include "SFML/Graphics/Rect.hpp"
00005 #include "SFML/System/Vector2.hpp"
00006 #include "animatedObject.hpp"
00007 #include "player.hpp"
00008 #include "point2d.hpp"
00009
00010 class GamePlayer : public Player {
00011 public:
00012
        static constexpr std::string_view s_spriteSheetPath{
00013
             "resource/player/spritesheet.png"};
00014
00015
        enum class AnimationType {
00016
          standing,
00017
00018
           moveEast,
00019
           moveSouth,
00020
           moveWest.
00021
           moveSpecial,
00022
           count
00023
00024
00025 public:
00026
        GamePlayer();
00027
00028
        virtual void animate() override;
        virtual void update() override;
00030
00031 private:
00032
         int m_aninmationFrameIndicator{};
00033
         float m_animationFrameDuration{.5};
00034
         float m_animationTimer{};
00035
         AnimationType m_animationType{AnimationType::standing};
00036
00037
         void nextAnimationFrame();
00038
         void updateTextureRect();
00039
00040
         void setAnimationType(AnimationType type);
00041
00042
         AnimationType getAnimationTypeOf(MoveDirection moveDirection) const;
00043
         int getFrameCountOfAnimation(AnimationType type) const;
00044
         sf::IntRect getCurrentAnimationFrameSpriteRectangle() const;
00045
         int getRowOfAnimation() const;
00046 };
00047
00048 #endif // GAMEPLAYER_H_
```

Index

```
\simAnimatedObject
                                                            m currentFrameId, 31
     AnimatedObject, 17
                                                            m position, 31
\simBush
                                                            move, 27
     Bush, 44
                                                            nextFrame, 28
\simEngine
                                                            setFrame, 29
    Engine, 53
                                                            setPosition, 29
\simGameObject
                                                       animatedSpriteSheet.cpp, 133
    GameObject, 86
                                                            operator>>, 133
                                                       animatedSpriteSheet.hpp, 135, 136
\simPlayer
     Player, 112
                                                       AnimatedSpriteSheet::AnimationFrameData, 32
                                                            m duration, 34
add
                                                            m position, 34
     Engine, 54, 55
                                                            m_size, 34
addRotationSpeed
                                                            operator>>, 33
    FidgetSpinner, 74
                                                            toIntRect, 33
addSomeRenderables
                                                       animateObjects
    main.cpp, 159
                                                            Engine, 55
animate
                                                       animation
    AnimatedObject, 17
                                                            main.cpp, 179
    AnimatedSpriteSheet, 22
                                                       animationData_t
    Bush, 44
                                                            AnimatedSpriteSheet, 21
    FidgetSpinner, 74
                                                            FidgetSpinner, 73
    GamePlayer, 92
                                                       AnimationType
     Player, 112
                                                            GamePlayer, 91
animatedObjecsCollection t
     Engine, 51
                                                       Ball. 34
AnimatedObject, 15
                                                            Ball, 36
                                                            getMoveVector, 36
    ~AnimatedObject, 17
    animate, 17
                                                            getPosition, 36
    getPosition, 17
                                                            isDead, 36
    m_position, 19
                                                            m_dead, 41
    move, 17
                                                            m_movementSpeed, 41
    setPosition, 18
                                                            m moveVectorBase, 41
animatedObject.cpp, 131
                                                            m position, 42
animatedObject.hpp, 132
                                                            m timeToLife, 42
AnimatedSpriteSheet, 19
                                                            move, 37
    animate, 22
                                                            setMovementSpeed, 38
    AnimatedSpriteSheet, 22
                                                            setMoveVectorBase, 38
                                                            setPosition, 39
    animationData_t, 21
    getCurrentAnimationFrameCount, 23
                                                            setPositon, 39
    getCurrentAnimationFrameData, 23
                                                            update, 40
    getCurrentAnimationFrameDuration, 24
                                                       ball.cpp, 180
    getCurrentAnimationFrameRect, 24
                                                       ball.hpp, 180, 181
    getPosition, 25
                                                       balls
    loadFrame, 25
                                                            main.cpp, 175
    loadFromConfigFile, 25
                                                       basePath
    loadSpritesheet, 27
                                                            G, 13
                                                       buildWindow
    m animationsData, 30
                                                            Engine, 56
    m animationTimer, 30
    m_currentAnimationTypeIndex, 31
                                                       Bush, 42
```

\sim Bush, 44	clear, 57
animate, 44	Clock, 51
Bush, 44	display, 57
getCurrentSpriteRectangle, 45	Drawable, 51
getPosition, 45	drawableCollection_t, 51
m_animationFrameDuration, 48	drawDrawables, 58
m_animationTimer, 48	Engine, 52
m_aninmationFrameIndicator, 48	Event, 51
m_position, 48	eventHandler_t, 52
move, 45	getInstance, 58
nextSprite, 46	getLastFrameDuration, 60
s_spriteSheetPath, 48	getResolution, 60
setPosition, 47	getWindow, 60
bush	handleEvents, 61
main.cpp, 175	loop, 62
bush.cpp, 182	m_animatedObjectsCollection, 69
bush.hpp, 183	m_clock, 69
5451pp, 100	m drawablesCollection, 69
clear	m_eventHandlers, 70
Engine, 57	m_lastFrameDuration, 70
Clock	m loopFunction, 70
Engine, 51	– •
cordinate t	m_maxFPS, 70
Point2d, 122	m_resoltuon, 70
count	m_window, 71
GamePlayer, 91, 92	m_windowTitle, 71
Player, 112	remove, 63
-	render, 63
customLoop	RenderWindow, 52
main.cpp, 164	s_instancePtr, 71
customLoopFunction	setEventHandler, 64
main.cpp, 160	setLoopFunction, 65
Depresented Liet 2	setMaxFps, 65
Deprecated List, 3	setResolution, 66, 67
display	setWindowTitle, 67
Engine, 57	Shape, 52
distanceTo	Time, 52
Point2d, 123	engine.cpp, 137
draw	engine.hpp, 139, 140
obstacle::LineSegment, 107	Event
Drawable	Engine, 51
Engine, 51	eventHandler_t
drawableCollection_t	Engine, 52
Engine, 51	
drawDrawables	FidgetSpinner, 72
Engine, 58	addRotationSpeed, 74
drwables	animate, 74
G, 13	animationData_t, 73
	FidgetSpinner, 74
east	getCenterPoint, 75
GamePlayer, 92	getCurrentAnimationFrameCount, 75
Player, 112	getCurrentAnimationFrameData, 76
eng	getCurrentAnimationFrameDuration, 76
main.cpp, 175	getCurrentAnimationFrameRect, 77
Engine, 49	getPosition, 77
\sim Engine, 53	loadFrame, 77
add, 54, 55	loadFromConfigFile, 78
animatedObjecsCollection_t, 51	loadSpritesheet, 79
animateObjects, 55	m_angle, 83
buildWindow, 56	m_animationsData, 83
	m_animadonoData, oo

m_animationTimer, 84	nextAnimationFrame, 99
m_currentAnimationTypeIndex, 84	north, 92
m_currentFrameId, 84	s_spriteSheetPath, 106
m_position, 84	setAnimationType, 100
m_rotationResistance, 84	setIsMoving, 100
m_rotationspeed, 85	setMovementSpeed, 101
move, 80	setPosition, 101
nextFrame, 81	south, 92
setFrame, 81	standing, 91
setPosition, 82	stopMoving, 102
setRotationResistance, 83	update, 103
fidgetSpinner	updateTextureRect, 104
main.cpp, 175	west, 92
fidgetSpinner.cpp, 184	gamePlayer.cpp, 186
fidgetSpinner.hpp, 185	gamePlayer.hpp, 188, 189
3	getAnimationTypeOf
G, 13	GamePlayer, 93
basePath, 13	getCenterPoint
drwables, 13	FidgetSpinner, 75
logstream, 13	getCurrentAnimationFrameCount
moveVertically, 13	AnimatedSpriteSheet, 23
GameObject, 85	FidgetSpinner, 75
~GameObject, 86	getCurrentAnimationFrameData
getPosition, 87	AnimatedSpriteSheet, 23
m_position, 88	FidgetSpinner, 76
move, 87	
setPosition, 87	getCurrentAnimationFrameDuration
GameObject.cpp, 141, 142	AnimatedSpriteSheet, 24
GameObject.hpp, 142, 143	FidgetSpinner, 76
GamePlayer, 89	getCurrentAnimationFrameRect
animate, 92	AnimatedSpriteSheet, 24
AnimationType, 91	FidgetSpinner, 77
count, 91, 92	getCurrentAnimationFrameSpriteRectangle
east, 92	GamePlayer, 93
GamePlayer, 92	getCurrentSpriteRectangle
getAnimationTypeOf, 93	Bush, 45
getCurrentAnimationFrameSpriteRectangle, 93	getEnd
getFrameCountOfAnimation, 94	obstacle::LineSegment, 107
•	getFrameCountOfAnimation
getMovementSpeed, 95	GamePlayer, 94
getMoveVector, 95	getInstance
getMoveVectorOrigin, 95	Engine, 58
getPosition, 96	getLastFrameDuration
getRowOfAnimation, 96	Engine, 60
isMoving, 97	getMovementSpeed
isMovingDiagonaly, 98	GamePlayer, 95
m_animationFrameDuration, 105	Player, 112
m_animationTimer, 105	getMoveVector
m_animationType, 105	Ball, 36
m_aninmationFrameIndicator, 105	GamePlayer, 95
m_isMoving, 105	Player, 113
m_movementSpeed, 105	getMoveVectorOrigin
m_position, 106	GamePlayer, 95
move, 98	Player, 113
MoveDirection, 91	getPosition
moveEast, 91	AnimatedObject, 17
moveNorth, 91	AnimatedSpriteSheet, 25
moveSouth, 91	Ball, 36
moveSpecial, 91	Bush, 45
moveWest. 91	Duoli, +0

FidgetSpinner, 77	AnimatedSpriteSheet, 27
GameObject, 87	FidgetSpinner, 79
GamePlayer, 96	log.cpp, 146
Player, 114	log.hpp, 147, 149
getRandomColor	LOGERROR, 147
main.cpp, 165	LOGINFO, 147
getResolution	LOGINFON, 148
Engine, 60	LOGTRACEN, 148
getRowOfAnimation	LOGWARN, 148
GamePlayer, 96	LOGWARNN, 148
getStart	LOGERROR
obstacle::LineSegment, 108	log.hpp, 147
getWindow	LOGINFO
Engine, 60	log.hpp, 147
getX	LOGINFON
Point2d, 124	log.hpp, 148
getY	logstream
Point2d, 124	G, 13
	LOGTRACEN
handleBalls	log.hpp, 148
main.cpp, 165	LOGWARN
handleEvents	log.hpp, 148
Engine, 61	LOGWARNN
handlePlayerMovement	
main.cpp, 166	log.hpp, 148
Пап.орр, 100	loop
initializeBush	Engine, 62
main.cpp, 167	m angle
initializeFidgetSpinner	FidgetSpinner, 83
main.cpp, 168	m_animatedObjectsCollection
initialziePlayer	Engine, 69
main.cpp, 168	m animationFrameDuration
isDead	Bush, 48
Ball, 36	•
isMoving	GamePlayer, 105 m animationsData
•	-
GamePlayer, 97	AnimatedSpriteSheet, 30
Player, 114, 115	FidgetSpinner, 83
isMovingDiagonaly	m_animationTimer
GamePlayer, 98	AnimatedSpriteSheet, 30
Player, 115	Bush, 48
keyPressedEventHandler	FidgetSpinner, 84
<u> </u>	GamePlayer, 105
main.cpp, 169	m_animationType
keyReleasedEventHandler	GamePlayer, 105
main.cpp, 170	$m_aninmationFrameIndicator$
longth	Bush, 48
length	GamePlayer, 105
Point2d, 124	m_clock
LineSegment	Engine, 69
obstacle::LineSegment, 107	m_color
lineSegment.cpp, 143, 144	obstacle::LineSegment, 109
lineSegment.hpp, 144, 145	m_currentAnimationTypeIndex
loadFrame	AnimatedSpriteSheet, 31
AnimatedSpriteSheet, 25	FidgetSpinner, 84
FidgetSpinner, 77	m currentFrameId
IoadFromConfigFile	AnimatedSpriteSheet, 31
AnimatedSpriteSheet, 25	FidgetSpinner, 84
FidgetSpinner, 78	m_dead
loadSpritesheet	Ball, 41
	- w., · ·

m_drawablesCollection	customLoop, 164
Engine, 69	customLoopFunction, 160
m_duration	eng, 175
AnimatedSpriteSheet::AnimationFrameData, 34	fidgetSpinner, 175
m_eventHandlers	getRandomColor, 165
Engine, 70	handleBalls, 165
m_isMoving	handlePlayerMovement, 166
GamePlayer, 105	initializeBush, 167
Player, 120	initializeFidgetSpinner, 168
m lastFrameDuration	initialziePlayer, 168
Engine, 70	keyPressedEventHandler, 169
m_loopFunction	keyReleasedEventHandler, 170
Engine, 70	main, 161, 171, 178
m maxFPS	player, 175
Engine, 70	setUpCustomEvents, 161
m_movementSpeed	spinTheFidget, 172
Ball, 41	· -
	throwBall, 173
GamePlayer, 105	throwBallPlayer, 174
Player, 120	move
m_moveVectorBase	AnimatedObject, 17
Ball, 41	AnimatedSpriteSheet, 27
m_points	Ball, 37
obstacle::LineSegment, 109	Bush, 45
m_position	FidgetSpinner, 80
AnimatedObject, 19	GameObject, 87
AnimatedSpriteSheet, 31	GamePlayer, 98
AnimatedSpriteSheet::AnimationFrameData, 34	Player, 116
Ball, 42	MoveDirection
Bush, 48	GamePlayer, 91
FidgetSpinner, 84	Player, 112
GameObject, 88	moveEast
GamePlayer, 106	GamePlayer, 91
Player, 120	moveNorth
m resoltuon	GamePlayer, 91
Engine, 70	moveSouth
m rotationResistance	GamePlayer, 91
FidgetSpinner, 84	moveSpecial
m_rotationspeed	GamePlayer, 91
FidgetSpinner, 85	moveVertically
	G, 13
m_size AnimatedSpriteSheet::AnimationFrameData, 34	moveWest
m timeToLife	GamePlayer, 91
_ '	Gameriayer, 91
Ball, 42	nextAnimationFrame
m_window	GamePlayer, 99
Engine, 71	-
m_windowTitle	nextFrame
Engine, 71	AnimatedSpriteSheet, 28
m_x	FidgetSpinner, 81
Point2d, 129	nextSprite
m_y	Bush, 46
Point2d, 129	north
main	GamePlayer, 92
main.cpp, 161, 171, 178	Player, 112
main.cpp, 158, 162, 163, 176, 178, 179	
addSomeRenderables, 159	obstacle, 14
animation, 179	obstacle::LineSegment, 106
balls, 175	draw, 107
bush, 175	getEnd, 107
, -	getStart, 108

LineSegment, 107	player.cpp, 149
m color, 109	player.hpp, 150, 151
m points, 109	Point2d, 121
—	•
setColor, 108	cordinate_t, 122
setEnd, 108	distanceTo, 123
setStart, 108	getX, 124
operator!=	getY, 124
Point2d, 127	length, 124
point2d.cpp, 153	m_x, 129
operator<<	
	m_y, 129
Point2d, 128	operator!=, 127
point2d.cpp, 154	operator<<, 128
operator>>	operator>>, 129
animatedSpriteSheet.cpp, 133	operator+, 128
AnimatedSpriteSheet::AnimationFrameData, 33	operator+=, 128
Point2d, 129	operator-, 128
point2d.cpp, 154	operator==, 129
operator+	operator*, 127, 128
Point2d, 128	•
•	Point2d, 122, 123
point2d.cpp, 153	setX, 125
operator+=	setY, 126
Point2d, 128	swap, 126
point2d.cpp, 153	toVector2f, 126
operator-	point2d.cpp, 152, 155
Point2d, 128	operator!=, 153
point2d.cpp, 153	operator<<, 154
	•
operator==	operator>>, 154
Point2d, 129	operator+, 153
point2d.cpp, 154	operator+=, 153
operator*	operator-, 153
Point2d, 127, 128	operator==, 154
point2d.cpp, 153	operator*, 153
	point2d.hpp, 156
Player, 110	F FF,
∼Player, 112	Readme, 1
animate, 112	README.md, 131
count, 112	
	remove
east, 112	Engine, 63
getMovementSpeed, 112	render
getMoveVector, 113	Engine, 63
getMoveVectorOrigin, 113	RenderWindow
getPosition, 114	Engine, 52
isMoving, 114, 115	
isMovingDiagonaly, 115	s instancePtr
m isMoving, 120	Engine, 71
m_movementSpeed, 120	s_spriteSheetPath
	Bush, 48
m_position, 120	
move, 116	GamePlayer, 106
MoveDirection, 112	setAnimationType
north, 112	GamePlayer, 100
setIsMoving, 117	setColor
setMovementSpeed, 117	obstacle::LineSegment, 108
setPosition, 118	setEnd
south, 112	obstacle::LineSegment, 108
stopMoving, 118, 119	setEventHandler
, -	Engine, 64
update, 119	_
west, 112	setFrame
player	AnimatedSpriteSheet, 29
main.cpp, 175	FidgetSpinner, 81

11 AA - 2	LUD
setIsMoving	toIntRect
GamePlayer, 100	AnimatedSpriteSheet::AnimationFrameData, 33
Player, 117	toVector2f
setLoopFunction	Point2d, 126
Engine, 65	
setMaxFps	update
Engine, 65	Ball, 40
setMovementSpeed	GamePlayer, 103
Ball, 38	Player, 119
GamePlayer, 101	UpdateableObject, 130
Player, 117	UpdateableObject, 130
setMoveVectorBase	update, 130
Ball, 38	updateableObject.cpp, 157
	updateableObject.hpp, 158
setPosition	
AnimatedObject, 18	updateTextureRect
AnimatedSpriteSheet, 29	GamePlayer, 104
Ball, 39	west
Bush, 47	west
FidgetSpinner, 82	GamePlayer, 92
GameObject, 87	Player, 112
GamePlayer, 101	
Player, 118	
setPositon	
Ball, 39	
setResolution	
Engine, 66, 67	
setRotationResistance	
FidgetSpinner, 83	
setStart	
obstacle::LineSegment, 108	
setUpCustomEvents	
main.cpp, 161	
setWindowTitle	
Engine, 67	
setX	
Point2d, 125	
setY	
Point2d, 126	
Shape	
Engine, 52	
south	
GamePlayer, 92	
Player, 112	
spinTheFidget	
main.cpp, 172	
standing	
GamePlayer, 91	
stopMoving	
GamePlayer, 102	
Player, 118, 119	
•	
swap	
Point2d, 126	
throwBall	
main.cpp, 173	
throwBallPlayer	
main.cpp, 174	
Time	
Engine, 52	