

RetroG24_C22_Centipede

v.3

Generated by Doxygen 1.11.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 BodyPart Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 BodyPart()	8
4.1.3 Member Function Documentation	8
4.1.3.1 addChild()	8
4.1.3.2 getItsChild()	8
4.1.3.3 getItsHitBox()	9
4.1.3.4 getItsParent()	9
4.1.3.5 getItsPosition()	9
4.1.3.6 getItsPreviousPosition()	9
4.1.3.7 getItsTarget()	9
4.1.3.8 getNextTarget()	9
4.1.3.9 setItsChild()	10
4.1.3.10 setItsHitBox()	10
4.1.3.11 setItsParent()	10
4.1.3.12 setItsPosition()	10
4.1.3.13 setItsTargetPos()	11
4.2 Bullet Class Reference	11
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	12
4.2.2.1 Bullet()	12
4.2.3 Member Function Documentation	12
4.2.3.1 getItsHitBox()	12
4.2.3.2 getItsPosition()	12
4.2.3.3 setItsHitBox()	12
4.2.3.4 setItsPosition()	13
4.2.3.5 updatePos()	13
4.3 Centipede Class Reference	13
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 Centipede()	14
4.3.3 Member Function Documentation	15

4.3.3.1	getItsDirection()	15
4.3.3.2	getItsHead()	15
4.3.3.3	getItsTail()	15
4.3.3.4	getNextPosition()	15
4.3.3.5	getWasMovingLeft()	16
4.3.3.6	getWasMovingRight()	16
4.3.3.7	hasReachedBottom()	16
4.3.3.8	isVerticalDirection()	16
4.3.3.9	setHasReachedBottom()	16
4.3.3.10	setItsDirection()	17
4.3.3.11	setItsTail()	17
4.3.3.12	setVerticalDirection()	17
4.3.3.13	setWasMovingLeft()	17
4.3.3.14	setWasMovingRight()	18
4.4	Direction Struct Reference	18
4.4.1	Detailed Description	18
4.4.2	Member Data Documentation	18
4.4.2.1	dirX	18
4.4.2.2	dirY	19
4.5	Game Class Reference	19
4.5.1	Detailed Description	20
4.5.2	Constructor & Destructor Documentation	21
4.5.2.1	Game()	21
4.5.3	Member Function Documentation	21
4.5.3.1	centipedeBoardCollision()	21
4.5.3.2	centipedeMushroomCollision()	21
4.5.3.3	centipedeToCentipedeCollision()	21
4.5.3.4	getCurrentLevel()	22
4.5.3.5	getIsPiercingActive()	22
4.5.3.6	getIsRafaleActive()	22
4.5.3.7	getItsBoard()	22
4.5.3.8	getItsBullets()	23
4.5.3.9	getItsCentipedes()	23
4.5.3.10	getItsMushrooms()	23
4.5.3.11	getItsPlayer()	23
4.5.3.12	getItsPowerups()	23
4.5.3.13	getItsScore()	24
4.5.3.14	getItsSpider()	24
4.5.3.15	isColliding() [1/5]	24
4.5.3.16	isColliding() [2/5]	24
4.5.3.17	isColliding() [3/5]	25
4.5.3.18	isColliding() [4/5]	25

4.5.3.19 isColliding() [5/5]	25
4.5.3.20 isGameLost()	26
4.5.3.21 isLevelWon()	26
4.5.3.22 movePlayer()	26
4.5.3.23 setBoard()	26
4.5.3.24 setItsPiercingActive()	26
4.5.3.25 setItsRafaleActive()	27
4.5.3.26 sliceCentipede()	27
4.6 Leaderboard Class Reference	27
4.6.1 Detailed Description	28
4.6.2 Constructor & Destructor Documentation	28
4.6.2.1 Leaderboard()	28
4.6.3 Member Function Documentation	28
4.6.3.1 addScore()	28
4.6.3.2 getItsBestScores()	28
4.7 Mushroom Class Reference	29
4.7.1 Detailed Description	29
4.7.2 Constructor & Destructor Documentation	29
4.7.2.1 Mushroom()	29
4.7.3 Member Function Documentation	30
4.7.3.1 getItsGridPosition()	30
4.7.3.2 getItsHitBox()	30
4.7.3.3 getItsState()	30
4.7.3.4 setItsGridPosition()	30
4.7.3.5 setItsHitBox()	30
4.8 Player Class Reference	31
4.8.1 Detailed Description	31
4.8.2 Constructor & Destructor Documentation	31
4.8.2.1 Player()	31
4.8.3 Member Function Documentation	32
4.8.3.1 getItsHitBox()	32
4.8.3.2 getItsHp()	32
4.8.3.3 getItsPosition()	32
4.8.3.4 setItsHitBox() [1/2]	32
4.8.3.5 setItsHitBox() [2/2]	33
4.8.3.6 setItsPosition()	33
4.8.3.7 updatePos()	33
4.9 Position Struct Reference	33
4.9.1 Detailed Description	34
4.9.2 Member Data Documentation	34
4.9.2.1 posX	34
4.9.2.2 posY	34

4.10 PowerUp Class Reference	34
4.10.1 Detailed Description	35
4.10.2 Constructor & Destructor Documentation	35
4.10.2.1 PowerUp()	35
4.10.3 Member Function Documentation	35
4.10.3.1 getItsHitbox()	35
4.10.3.2 getItsPosition()	35
4.10.3.3 getItsType()	35
4.10.3.4 setItsHitbox()	35
4.10.3.5 setItsPosition()	36
4.11 Spider Class Reference	36
4.11.1 Detailed Description	37
4.11.2 Constructor & Destructor Documentation	37
4.11.2.1 Spider()	37
4.11.3 Member Function Documentation	37
4.11.3.1 getItsDirection()	37
4.11.3.2 getItsHitBox()	37
4.11.3.3 getItsHorizontaleDirection()	38
4.11.3.4 move()	38
4.11.3.5 setItsDirection()	38
4.11.3.6 setItsHitBox()	38
4.11.3.7 setItsHorizontaleDirection()	38
4.12 Widget Class Reference	39
4.12.1 Detailed Description	40
4.12.2 Constructor & Destructor Documentation	40
4.12.2.1 Widget()	40
4.12.3 Member Function Documentation	40
4.12.3.1 drawBullet()	40
4.12.3.2 drawCentipede()	40
4.12.3.3 drawHeadUpDisplay()	41
4.12.3.4 drawMushrooms()	41
4.12.3.5 drawPlayer()	41
4.12.3.6 drawPowerUps()	41
4.12.3.7 drawSpider()	42
4.12.3.8 keyPressEvent()	42
4.12.3.9 keyReleaseEvent()	42
4.12.3.10 paintEvent()	42
4.12.3.11 pauseGame()	43
4.12.3.12 resizeEvent()	43
5 File Documentation	45
5.1 bodypart.h File Reference	45

5.1.1 Detailed Description	45
5.2 bodypart.h	45
5.3 bullet.h File Reference	46
5.3.1 Detailed Description	46
5.4 bullet.h	46
5.5 centipede.h File Reference	47
5.5.1 Detailed Description	47
5.6 centipede.h	47
5.7 game.h File Reference	48
5.7.1 Detailed Description	48
5.8 game.h	48
5.9 leaderboard.h File Reference	50
5.9.1 Detailed Description	50
5.10 leaderboard.h	50
5.11 mushroom.h File Reference	51
5.11.1 Detailed Description	51
5.12 mushroom.h	51
5.13 player.h File Reference	51
5.13.1 Detailed Description	52
5.14 player.h	52
5.15 powerup.h File Reference	52
5.15.1 Detailed Description	53
5.15.2 Enumeration Type Documentation	53
5.15.2.1 powerupType	53
5.16 powerup.h	53
5.17 spider.h File Reference	54
5.17.1 Detailed Description	54
5.18 spider.h	54
5.19 typeDef.h File Reference	54
5.19.1 Detailed Description	55
5.19.2 Variable Documentation	55
5.19.2.1 BOARD_HEIGHT	55
5.19.2.2 BOARD_WIDTH	55
5.19.2.3 BULLET_SPEED	56
5.19.2.4 CENTIPEDE_BODYPART_SIZE	56
5.19.2.5 CENTIPEDE_LENGTH	56
5.19.2.6 CENTIPEDE_SPAWN_XPOS	56
5.19.2.7 CENTIPEDE_SPAWN_YPOS	56
5.19.2.8 INCREMENT_INTERVAL	56
5.19.2.9 MUSHROOM_SIZE	56
5.19.2.10 MUSHROOMS_AMOUNT	56
5.19.2.11 PLAYER_SIZE	57

5.19.2.12 PLAYER_SPEED	57
5.19.2.13 POWERUP_DROP_RATE	57
5.19.2.14 POWERUP_PIERCING_DURATION	57
5.19.2.15 POWERUP_RAFALE_DURATION	57
5.19.2.16 POWERUP_RAFALE_FIRERATE	57
5.19.2.17 SAVEFILE_NAME	57
5.19.2.18 SHOW_HITBOXES	57
5.19.2.19 SPIDER_SPEED	58
5.20 typeDef.h	58
5.21 widget.h File Reference	58
5.21.1 Detailed Description	59
5.22 widget.h	59
Index	61

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

BodyPart	7
Bullet	11
Centipede	13
Direction	18
Game	19
Leaderboard	27
Mushroom	29
Player	31
Position	33
PowerUp	34
QWidget	
Widget	39
Spider	36

Chapter 2

Class Index

2.1 Class List

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

BodyPart	Represents a part of the centipede's body	7
Bullet	Represents a bullet in the game	11
Centipede	Represents a centipede entity in a game	13
Direction	Structure representing a direction with x and y components	18
Game	Class representing the game management	19
Leaderboard	Manages and stores high scores in a leaderboard	27
Mushroom	Class representing a mushroom in the game	29
Player	Class representing the player in the game	31
Position	Structure representing a position with x and y coordinates	33
PowerUp	Represents a power-up in the game with a hitbox and type	34
Spider	Represents a spider in the game with a hitbox and movement capabilities	36
Widget	Class representing the main game widget	39

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

bodypart.h	Defines the BodyPart (p. 7) class, which represents a part of the centipede's body	45
bullet.h	Defines the Bullet (p. 11) class, which represents a bullet in the game	46
centipede.h	Defines the Centipede (p. 13) class, which represents a centipede entity in a game	47
game.h	Defines the Game (p. 19) class, which represents the game management	48
leaderboard.h	Defines the Leaderboard (p. 27) class, which manages high scores in the game	50
mushroom.h	Defines the Mushroom (p. 29) class, which represents a mushroom in the game	51
player.h	Defines the Player (p. 31) class, which represents the player in the game	51
powerup.h	Header file for the PowerUp (p. 34) class	52
spider.h	Header file for the Spider (p. 36) class	54
typeDef.h	Defines common typedefs and constants used in the game	54
widget.h	Defines the Widget (p. 39) class, which represents the main game widget	58

Chapter 4

Class Documentation

4.1 BodyPart Class Reference

Represents a part of the centipede's body.

```
#include <bodypart.h>
```

Public Member Functions

- **BodyPart** (int size)
*Constructor for the **BodyPart** (p. 7) class.*
- **~BodyPart** ()
*Destructor for the **BodyPart** (p. 7) class.*
- **Position** **getItsPosition** ()
Gets the position of the body part.
- **Position** **getItsPreviousPosition** ()
Gets the previous position of the body part.
- **BodyPart *** **getItsChild** ()
Gets the child body part.
- **BodyPart *** **getItsParent** ()
Gets the parent body part.
- **QRect** **getItsHitBox** ()
Gets the hitbox of the body part.
- void **setItsPosition** (**Position** pos)
Sets the position of the body part.
- void **setItsChild** (**BodyPart ***child)
Sets the child body part.
- void **setItsParent** (**BodyPart ***parent)
Sets the parent body part.
- void **setItsHitBox** (QRect hitbox)
Sets the hitbox of the body part.
- void **setItsTargetPos** (**Position** targetPos)
Sets the target position for the body part's movement.
- void **updatePos** ()
Updates the position of the body part based on its target position.
- **Position** **getNextTarget** (**Direction** centipedeDir, int caseLength)
Calculates the next target position based on the centipede's direction and movement distance.
- **Position** **getItsTarget** ()
Gets the target position of the body part.
- void **addChild** (**BodyPart ***child)
addChild add a new bodypart child to the bodypart

4.1.1 Detailed Description

Represents a part of the centipede's body.

The **BodyPart** (p. 7) class manages the individual segments of the centipede, including their position, hitbox, and relationships to other segments.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BodyPart()

```
BodyPart::BodyPart (
    int size)
```

Constructor for the **BodyPart** (p. 7) class.

Parameters

<i>size</i>	The size of the body part.
-------------	----------------------------

4.1.3 Member Function Documentation

4.1.3.1 addChild()

```
void BodyPart::addChild (
    BodyPart * child)
```

addChild add a new bodypart child to the bodypart

Parameters

<i>child</i>	is his new bodypart
--------------	---------------------

4.1.3.2 getItsChild()

```
BodyPart * BodyPart::getItsChild ()
```

Gets the child body part.

Returns

Pointer to the child body part.

4.1.3.3 getItsHitBox()

```
QRect BodyPart::getItsHitBox ()
```

Gets the hitbox of the body part.

Returns

The hitbox of the body part as a QRect.

4.1.3.4 getItsParent()

```
BodyPart * BodyPart::getItsParent ()
```

Gets the parent body part.

Returns

Pointer to the parent body part.

4.1.3.5 getItsPosition()

```
Position BodyPart::getItsPosition ()
```

Gets the position of the body part.

Returns

The position of the body part.

4.1.3.6 getItsPreviousPosition()

```
Position BodyPart::getItsPreviousPosition ()
```

Gets the previous position of the body part.

Returns

The previous position of the body part.

4.1.3.7 getItsTarget()

```
Position BodyPart::getItsTarget ()
```

Gets the target position of the body part.

Returns

The target position of the body part.

4.1.3.8 getNextTarget()

```
Position BodyPart::getNextTarget (
    Direction centipedeDir,
    int caseLength)
```

Calculates the next target position based on the centipede's direction and movement distance.

Parameters

<i>centipedeDir</i>	The direction of movement for the centipede.
<i>caseLength</i>	The distance to move.

Returns

The next target position.

4.1.3.9 setItsChild()

```
void BodyPart::setItsChild (  
    BodyPart * child)
```

Sets the child body part.

Parameters

<i>child</i>	Pointer to the child body part.
--------------	---------------------------------

4.1.3.10 setItsHitBox()

```
void BodyPart::setItsHitBox (  
    QRect hitbox)
```

Sets the hitbox of the body part.

Parameters

<i>hitbox</i>	The new hitbox of the body part.
---------------	----------------------------------

4.1.3.11 setItsParent()

```
void BodyPart::setItsParent (  
    BodyPart * parent)
```

Sets the parent body part.

Parameters

<i>parent</i>	Pointer to the parent body part.
---------------	----------------------------------

4.1.3.12 setItsPosition()

```
void BodyPart::setItsPosition (  
    Position pos)
```

Sets the position of the body part.

Parameters

<i>pos</i>	The new position of the body part.
------------	------------------------------------

4.1.3.13 setItsTargetPos()

```
void BodyPart::setItsTargetPos (
    Position targetPos)
```

Sets the target position for the body part's movement.

Parameters

<i>targetPos</i>	The new target position.
------------------	--------------------------

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

- **bodypart.h**
- bodypart.cpp

4.2 Bullet Class Reference

Represents a bullet in the game.

```
#include <bullet.h>
```

Public Member Functions

- **Bullet** (int x, int y, int size)
*Constructor for the **Bullet** (p. 11) class.*
- **~Bullet** ()
*Destructor for the **Bullet** (p. 11) class.*
- void **updatePos** ()
Updates the position of the bullet.
- **QRect** **getItsHitBox** ()
Gets the hitbox of the bullet.
- **Position** **getItsPosition** ()
Gets the position of the bullet.
- void **setItsPosition** (**Position** position)
Sets the position of the bullet.
- void **setItsHitBox** (QRect hitbox)
Sets the hitbox of the bullet.

4.2.1 Detailed Description

Represents a bullet in the game.

The **Bullet** (p. 11) class manages the properties and behavior of a bullet, including its position and hitbox.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Bullet()

```
Bullet::Bullet (
    int x,
    int y,
    int size)
```

Constructor for the **Bullet** (p. 11) class.

Parameters

<i>x</i>	The x-coordinate of the bullet's initial position.
<i>y</i>	The y-coordinate of the bullet's initial position.
<i>size</i>	The size of the bullet.

4.2.3 Member Function Documentation

4.2.3.1 getItsHitBox()

```
QRect Bullet::getItsHitBox ()
```

Gets the hitbox of the bullet.

Returns

The hitbox of the bullet as a QRect.

4.2.3.2 getItsPosition()

```
Position Bullet::getItsPosition ()
```

Gets the position of the bullet.

Returns

The position of the bullet as a **Position** (p. 33) struct.

4.2.3.3 setItsHitBox()

```
void Bullet::setItsHitBox (
    QRect hitbox)
```

Sets the hitbox of the bullet.

Parameters

<i>hitbox</i>	The new hitbox of the bullet.
---------------	-------------------------------

4.2.3.4 setItsPosition()

```
void Bullet::setItsPosition (  
    Position position)
```

Sets the position of the bullet.

Parameters

<i>position</i>	The new position of the bullet.
-----------------	---------------------------------

4.2.3.5 updatePos()

```
void Bullet::updatePos ()
```

Updates the position of the bullet.

This function updates the position of the bullet based on its current direction and speed.

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

- **bullet.h**
- **bullet.cpp**

4.3 Centipede Class Reference

Represents a centipede entity in a game.

```
#include <centipede.h>
```

Public Member Functions

- **Centipede** (**BodyPart** *head)
*Constructs a new **Centipede** (p. 13) object.*
- **~Centipede** ()
*Destroys the **Centipede** (p. 13) object.*
- **BodyPart** * **getItsHead** ()
*Gets a pointer to the head **BodyPart** (p. 7) of the centipede.*
- **BodyPart** * **getItsTail** ()
*Gets a pointer to the tail **BodyPart** (p. 7) of the centipede.*
- void **setItsTail** (**BodyPart** *tail)
*Sets the tail **BodyPart** (p. 7) of the centipede.*
- void **setItsDirection** (**Direction** dir)
Sets the direction of movement for the centipede.
- **Position** **getNextPosition** (int distance)
Calculates the next position of the centipede based on the current direction and distance to move.
- **Direction** **getItsDirection** ()
Gets the direction of movement for the centipede.
- bool **hasReachedBottom** ()
Checks if the centipede has reached the bottom of the screen.
- bool **isVerticalDirection** ()
Checks if the centipede is moving in a vertical direction (up or down).
- void **setVerticalDirection** (bool isCentipedeGoingDown)
Sets the vertical direction of movement for the centipede.
- void **setHasReachedBottom** (bool **hasReachedBottom**)
Sets the flag indicating whether the centipede has reached the bottom of the screen.
- void **setWasMovingRight** (bool value)
Sets the flag indicating whether the centipede was moving right.
- void **setWasMovingLeft** (bool value)
Sets the flag indicating whether the centipede was moving left.
- bool **getWasMovingRight** ()
Gets the flag indicating whether the centipede was moving right.
- bool **getWasMovingLeft** ()
Gets the flag indicating whether the centipede was moving left.

4.3.1 Detailed Description

Represents a centipede entity in a game.

This class manages the movement and characteristics of a centipede, including its head, tail, movement direction, and state flags.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Centipede()

```
Centipede::Centipede (
    BodyPart * head)
```

Constructs a new **Centipede** (p. 13) object.

Parameters

<i>head</i>	Pointer to the head BodyPart (p. 7) of the centipede.
-------------	--

4.3.3 Member Function Documentation

4.3.3.1 getItsDirection()

```
Direction Centipede::getItsDirection ()
```

Gets the direction of movement for the centipede.

Returns

The direction of movement for the centipede.

4.3.3.2 getItsHead()

```
BodyPart * Centipede::getItsHead ()
```

Gets a pointer to the head **BodyPart** (p. 7) of the centipede.

Returns

Pointer to the head **BodyPart** (p. 7) of the centipede.

4.3.3.3 getItsTail()

```
BodyPart * Centipede::getItsTail ()
```

Gets a pointer to the tail **BodyPart** (p. 7) of the centipede.

Returns

Pointer to the tail **BodyPart** (p. 7) of the centipede.

4.3.3.4 getNextPosition()

```
Position Centipede::getNextPosition (  
    int distance)
```

Calculates the next position of the centipede based on the current direction and distance to move.

Parameters

<i>distance</i>	The distance to move the centipede.
-----------------	-------------------------------------

Returns

The calculated next position of the centipede.

4.3.3.5 getWasMovingLeft()

```
bool Centipede::getWasMovingLeft ()
```

Gets the flag indicating whether the centipede was moving left.

Returns

True if the centipede was moving left, otherwise false.

4.3.3.6 getWasMovingRight()

```
bool Centipede::getWasMovingRight ()
```

Gets the flag indicating whether the centipede was moving right.

Returns

True if the centipede was moving right, otherwise false.

4.3.3.7 hasReachedBottom()

```
bool Centipede::hasReachedBottom ()
```

Checks if the centipede has reached the bottom of the screen.

Returns

True if the centipede has reached the bottom of the screen, otherwise false.

4.3.3.8 isVerticalDirection()

```
bool Centipede::isVerticalDirection ()
```

Checks if the centipede is moving in a vertical direction (up or down).

Returns

True if the centipede is moving down, otherwise false.

4.3.3.9 setHasReachedBottom()

```
void Centipede::setHasReachedBottom (  
    bool hasReachedBottom)
```

Sets the flag indicating whether the centipede has reached the bottom of the screen.

Parameters

<i>hasReachedBottom</i>	True if the centipede has reached the bottom, otherwise false.
-------------------------	--

4.3.3.10 setItsDirection()

```
void Centipede::setItsDirection (  
    Direction dir)
```

Sets the direction of movement for the centipede.

Parameters

<i>dir</i>	The direction of movement for the centipede.
------------	--

4.3.3.11 setItsTail()

```
void Centipede::setItsTail (  
    BodyPart * tail)
```

Sets the tail **BodyPart** (p. 7) of the centipede.

Parameters

<i>tail</i>	Pointer to the tail BodyPart (p. 7) of the centipede.
-------------	--

4.3.3.12 setVerticalDirection()

```
void Centipede::setVerticalDirection (  
    bool isCentipedeGoingDown)
```

Sets the vertical direction of movement for the centipede.

Parameters

<i>isCentipedeGoingDown</i>	True if the centipede is moving down, otherwise false.
-----------------------------	--

4.3.3.13 setWasMovingLeft()

```
void Centipede::setWasMovingLeft (  
    bool value)
```

Sets the flag indicating whether the centipede was moving left.

Parameters

<i>value</i>	True if the centipede was moving left, otherwise false.
--------------	---

4.3.3.14 setWasMovingRight()

```
void Centipede::setWasMovingRight (
    bool value)
```

Sets the flag indicating whether the centipede was moving right.

Parameters

<i>value</i>	True if the centipede was moving right, otherwise false.
--------------	--

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

- **centipede.h**
- centipede.cpp

4.4 Direction Struct Reference

Structure representing a direction with x and y components.

```
#include <typeDef.h>
```

Public Attributes

- int **dirX**
- int **dirY**

4.4.1 Detailed Description

Structure representing a direction with x and y components.

4.4.2 Member Data Documentation**4.4.2.1 dirX**

```
int Direction::dirX
```

The x-component of the direction

4.4.2.2 dirY

```
int Direction::dirY
```

The y-component of the direction

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

- **typeDef.h**

4.5 Game Class Reference

Class representing the game management.

```
#include <game.h>
```

Public Member Functions

- **Game** (QRect board)
*Constructor for the **Game** (p. 19) class.*
- **~Game** ()
*Destructor for the **Game** (p. 19) class.*
- void **spawnCentipede** ()
Spawns a centipede.
- void **createMushrooms** ()
Creates mushrooms on the game board.
- void **shoot** ()
Shoots a bullet.
- void **moveBullets** ()
Moves the bullets.
- bool **isColliding** (**Mushroom** *mushroom, **Player** *player)
Checks for collision between a mushroom and the player.
- bool **isColliding** (**Mushroom** *mushroom, **Bullet** *bullet)
Checks for collision between a mushroom and a bullet.
- bool **isColliding** (**Centipede** *centipede, **Bullet** *bullet)
Checks for collision between a centipede and a bullet.
- bool **isColliding** (**Centipede** *centipede, **Mushroom** *mushroom)
Checks for collision between a centipede and a mushroom.
- bool **isColliding** (QRect hitbox1, QRect hitbox2)
Checks for collision between two rectangles.
- void **checkCollisions** ()
Checks for and handles all collisions in the game.
- void **sliceCentipede** (**BodyPart** *hittedPart, **Centipede** *centipede)
Slices a centipede when hit.
- void **movePlayer** (**Direction** &direction)
Moves the player in a specified direction.
- void **movePowerUps** ()
Moves powerups.
- std::vector< **Centipede** * > * **getItsCentipedes** ()

- Gets the vector of centipedes.*
 - `std::vector< Mushroom * > * getItsMushrooms ()`
- Gets the vector of mushrooms.*
 - `std::vector< Bullet * > getItsBullets ()`
- Gets the bullet vector.*
 - `Player * getItsPlayer ()`
- Gets the player.*
 - `int getItsScore ()`
- Gets the current game score.*
 - `QRect getItsBoard ()`
- Gets the game board rectangle.*
 - `std::vector< PowerUp * > getItsPowerups ()`
- Gets the vector of powerups.*
 - `int getCurrentLevel ()`
- Gets the current game level.*
 - `void setBoard (QRect board)`
- Sets the game board rectangle.*
 - `bool centipedeMushroomCollision (Centipede *centipede)`
- Checks for collision between a centipede and a mushroom.*
 - `bool centipedeToCentipedeCollision (Centipede *centipede)`
- Manages collision between two centipedes.*
 - `bool isLevelWon ()`
- Checks if the game has been won.*
 - `bool isGameLosed ()`
- Checks if the game has been lost.*
 - `bool getItsRafaleActive ()`
- Checks if the 'rafale' powerup is active.*
 - `bool getItsPiercingActive ()`
- Checks if the 'transpercant' powerup is active.*
 - `void setItsRafaleActive (bool isActive)`
- Sets the state of the 'rafale' powerup.*
 - `void setItsPiercingActive (bool isActive)`
- Sets the state of the 'transpercant' powerup.*
 - `void moveCentipede ()`
- Moves the centipede.*
 - `bool centipedeBoardCollision (Centipede *centipede, QRect board)`
- Checks for collision between a centipede and the game board.*
 - `void createSpider ()`
- Creates a spider in the game.*
 - `Spider * getItsSpider ()`
- Gets the spider in the game.*
 - `void moveSpider ()`
- Moves the spider in the game.*

4.5.1 Detailed Description

Class representing the game management.

This class manages the main elements of the game such as centipedes, mushrooms, the player, and the interactions between them.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Game()

```
Game::Game (
    QRect board)
```

Constructor for the **Game** (p. 19) class.

Parameters

<i>board</i>	Rectangle representing the game board.
--------------	--

4.5.3 Member Function Documentation

4.5.3.1 centipedeBoardCollision()

```
bool Game::centipedeBoardCollision (
    Centipede * centipede,
    QRect board)
```

Checks for collision between a centipede and the game board.

Parameters

<i>centipede</i>	Pointer to the centipede.
<i>board</i>	The game board rectangle.

Returns

True if a collision is detected, otherwise false.

4.5.3.2 centipedeMushroomCollision()

```
bool Game::centipedeMushroomCollision (
    Centipede * centipede)
```

Checks for collision between a centipede and a mushroom.

Parameters

<i>centipede</i>	Pointer to the centipede.
------------------	---------------------------

Returns

True if a collision is detected, otherwise false.

4.5.3.3 centipedeToCentipedeCollision()

```
bool Game::centipedeToCentipedeCollision (
    Centipede * centipede)
```

Manages collision between two centipedes.

Parameters

<i>centipede</i>	Pointer to the centipede.
------------------	---------------------------

Returns

True if there is a collision, otherwise false.

4.5.3.4 getCurrentLevel()

```
int Game::getCurrentLevel ()
```

Gets the current game level.

Returns

The current level of the game.

4.5.3.5 getIsPiercingActive()

```
bool Game::getIsPiercingActive ()
```

Checks if the 'transpercant' powerup is active.

Returns

True if the 'transpercant' powerup is active, otherwise false.

4.5.3.6 getIsRafaleActive()

```
bool Game::getIsRafaleActive ()
```

Checks if the 'rafale' powerup is active.

Returns

True if the 'rafale' powerup is active, otherwise false.

4.5.3.7 getItsBoard()

```
QRect Game::getItsBoard ()
```

Gets the game board rectangle.

Returns

The game board rectangle.

4.5.3.8 getItsBullets()

```
vector< Bullet * > Game::getItsBullets ()
```

Gets the bullet vector.

Returns

Vector containing all bullets.

4.5.3.9 getItsCentipedes()

```
std::vector< Centipede * > * Game::getItsCentipedes ()
```

Gets the vector of centipedes.

Returns

Pointer to the vector of centipedes.

4.5.3.10 getItsMushrooms()

```
std::vector< Mushroom * > * Game::getItsMushrooms ()
```

Gets the vector of mushrooms.

Returns

Pointer to the vector of mushrooms.

4.5.3.11 getItsPlayer()

```
Player * Game::getItsPlayer ()
```

Gets the player.

Returns

Pointer to the player.

4.5.3.12 getItsPowerups()

```
std::vector< PowerUp * > Game::getItsPowerups ()
```

Gets the vector of powerups.

Returns

The vector containing all powerups.

4.5.3.13 getItsScore()

```
int Game::getItsScore ()
```

Gets the current game score.

Returns

The game score.

4.5.3.14 getItsSpider()

```
Spider * Game::getItsSpider ()
```

Gets the spider in the game.

Returns

Pointer to the spider.

4.5.3.15 isColliding() [1/5]

```
bool Game::isColliding (  
    Centipede * centipede,  
    Bullet * bullet)
```

Checks for collision between a centipede and a bullet.

Parameters

<i>centipede</i>	Pointer to the centipede.
<i>bullet</i>	Pointer to the bullet.

Returns

True if a collision is detected, otherwise false.

4.5.3.16 isColliding() [2/5]

```
bool Game::isColliding (  
    Centipede * centipede,  
    Mushroom * mushroom)
```

Checks for collision between a centipede and a mushroom.

Parameters

<i>centipede</i>	Pointer to the centipede.
<i>mushroom</i>	Pointer to the mushroom.

Returns

True if a collision is detected, otherwise false.

4.5.3.17 isColliding() [3/5]

```
bool Game::isColliding (
    Mushroom * mushroom,
    Bullet * bullet)
```

Checks for collision between a mushroom and a bullet.

Parameters

<i>mushroom</i>	Pointer to the mushroom.
<i>bullet</i>	Pointer to the bullet.

Returns

True if a collision is detected, otherwise false.

4.5.3.18 isColliding() [4/5]

```
bool Game::isColliding (
    Mushroom * mushroom,
    Player * player)
```

Checks for collision between a mushroom and the player.

Parameters

<i>mushroom</i>	Pointer to the mushroom.
<i>player</i>	Pointer to the player.

Returns

True if a collision is detected, otherwise false.

4.5.3.19 isColliding() [5/5]

```
bool Game::isColliding (
    QRect hitbox1,
    QRect hitbox2)
```

Checks for collision between two rectangles.

Parameters

<i>hitbox1</i>	The first rectangle.
<i>hitbox2</i>	The second rectangle.

Returns

True if a collision is detected, otherwise false.

4.5.3.20 isGameLosed()

```
bool Game::isGameLosed ()
```

Checks if the game has been lost.

Returns

True if the game has been lost, otherwise false.

4.5.3.21 isLevelWon()

```
bool Game::isLevelWon ()
```

Checks if the game has been won.

Returns

True if the game has been won, otherwise false.

4.5.3.22 movePlayer()

```
void Game::movePlayer (  
    Direction & direction)
```

Moves the player in a specified direction.

Parameters

<i>direction</i>	Reference to the direction of movement.
------------------	---

4.5.3.23 setBoard()

```
void Game::setBoard (  
    QRect board)
```

Sets the game board rectangle.

Parameters

<i>board</i>	The new game board rectangle.
--------------	-------------------------------

4.5.3.24 setIsPiercingActive()

```
void Game::setIsPiercingActive (  
    bool isActive)
```

Sets the state of the 'transpercant' powerup.

Parameters

<i>isActive</i>	The state to set for the 'transpercant' powerup.
-----------------	--

4.5.3.25 setIsRafaleActive()

```
void Game::setIsRafaleActive (
    bool isActive)
```

Sets the state of the 'rafale' powerup.

Parameters

<i>isActive</i>	The state to set for the 'rafale' powerup.
-----------------	--

4.5.3.26 sliceCentipede()

```
void Game::sliceCentipede (
    BodyPart * hitPart,
    Centipede * centipede)
```

Slices a centipede when hit.

Parameters

<i>hitPart</i>	Pointer to the hit body part of the centipede.
<i>centipede</i>	Pointer to the centipede hit.

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

- **game.h**
- **game.cpp**

4.6 Leaderboard Class Reference

Manages and stores high scores in a leaderboard.

```
#include <leaderboard.h>
```

Public Member Functions

- **Leaderboard** (string filename)
*Constructor for the **Leaderboard** (p. 27) class.*
- map< string, int > **getItsBestScores** ()
Retrieves the best scores from the leaderboard.
- void **addScore** (int newScore, string username)
Adds a new score to the leaderboard.
- void **extract** ()
Extracts high scores from the file and updates the leaderboard.
- void **save** ()
Saves the current leaderboard data to the file.

4.6.1 Detailed Description

Manages and stores high scores in a leaderboard.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Leaderboard()

```
Leaderboard::Leaderboard (  
    string filename)
```

Constructor for the **Leaderboard** (p. 27) class.

Parameters

<i>filename</i>	The filename to load and save the leaderboard data.
-----------------	---

4.6.3 Member Function Documentation

4.6.3.1 addScore()

```
void Leaderboard::addScore (  
    int newScore,  
    string username)
```

Adds a new score to the leaderboard.

Parameters

<i>newScore</i>	The new score to add.
<i>username</i>	The username associated with the score.

4.6.3.2 getItsBestScores()

```
map< string, int > Leaderboard::getItsBestScores ()
```

Retrieves the best scores from the leaderboard.

Returns

A map containing usernames and their corresponding best scores.

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

- **leaderboard.h**
- **leaderboard.cpp**

4.7 Mushroom Class Reference

Class representing a mushroom in the game.

```
#include <mushroom.h>
```

Public Member Functions

- **Mushroom** (int x, int y, int size, **Position** gridPosition)
*Constructor for the **Mushroom** (p. 29) class.*
- **~Mushroom** ()
*Destructor for the **Mushroom** (p. 29) class.*
- void **damage** ()
Damages the mushroom.
- int **getItsState** ()
Gets the state of the mushroom.
- QRect **getItsHitBox** ()
Gets the hit box of the mushroom.
- **Position** **getItsGridPosition** ()
Gets the grid position of the mushroom.
- void **setItsHitBox** (QRect hitBox)
Sets the hit box of the mushroom.
- void **setItsGridPosition** (**Position** position)
Sets the grid position of the mushroom.

4.7.1 Detailed Description

Class representing a mushroom in the game.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Mushroom()

```
Mushroom::Mushroom (
    int x,
    int y,
    int size,
    Position gridPosition)
```

Constructor for the **Mushroom** (p. 29) class.

Parameters

<i>x</i>	The x-coordinate of the mushroom.
<i>y</i>	The y-coordinate of the mushroom.
<i>size</i>	The size of the mushroom.
<i>gridPosition</i>	The grid position of the mushroom.

4.7.3 Member Function Documentation

4.7.3.1 `getItsGridPosition()`

```
Position Mushroom::getItsGridPosition ()
```

Gets the grid position of the mushroom.

Returns

The grid position of the mushroom.

4.7.3.2 `getItsHitBox()`

```
QRect Mushroom::getItsHitBox ()
```

Gets the hit box of the mushroom.

Returns

The hit box of the mushroom.

4.7.3.3 `getItsState()`

```
int Mushroom::getItsState ()
```

Gets the state of the mushroom.

Returns

The state of the mushroom.

4.7.3.4 `setItsGridPosition()`

```
void Mushroom::setItsGridPosition (  
    Position position)
```

Sets the grid position of the mushroom.

Parameters

<i>position</i>	The new grid position of the mushroom.
-----------------	--

4.7.3.5 `setItsHitBox()`

```
void Mushroom::setItsHitBox (  
    QRect hitBox)
```

Sets the hit box of the mushroom.

Parameters

<i>hitBox</i>	The new hit box of the mushroom.
---------------	----------------------------------

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

- **mushroom.h**
- mushroom.cpp

4.8 Player Class Reference

Class representing the player in the game.

```
#include <player.h>
```

Public Member Functions

- **Player** (**Position** position, int size)
*Constructor for the **Player** (p. 31) class.*
- **~Player** ()
*Destructor for the **Player** (p. 31) class.*
- void **updatePos** (**Direction** direction)
Updates the position of the player based on the given direction.
- void **hit** ()
Decreases the hit points of the player when hit.
- **Position** **getItsPosition** ()
Gets the position of the player.
- int **getItsHp** ()
Gets the hit points of the player.
- **QRect** **getItsHitBox** ()
Gets the hit box of the player.
- void **setItsPosition** (**Position** position)
Sets the position of the player.
- void **setItsHitBox** (**QRect** hitBox)
Sets the hit box of the player.
- void **setItsHitBox** (**Position** position)
Sets the hit box of the player based on the given position.

4.8.1 Detailed Description

Class representing the player in the game.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Player()

```
Player::Player (
    Position position,
    int size)
```

Constructor for the **Player** (p. 31) class.

Parameters

<i>position</i>	The initial position of the player.
<i>size</i>	The size of the player.

4.8.3 Member Function Documentation

4.8.3.1 getItsHitBox()

```
QRect Player::getItsHitBox ()
```

Gets the hit box of the player.

Returns

The hit box of the player.

4.8.3.2 getItsHp()

```
int Player::getItsHp ()
```

Gets the hit points of the player.

Returns

The hit points of the player.

4.8.3.3 getItsPosition()

```
Position Player::getItsPosition ()
```

Gets the position of the player.

Returns

The position of the player.

4.8.3.4 setItsHitBox() [1/2]

```
void Player::setItsHitBox (  
    Position position)
```

Sets the hit box of the player based on the given position.

Parameters

<i>position</i>	The new position of the player.
-----------------	---------------------------------

4.8.3.5 setItsHitBox() [2/2]

```
void Player::setItsHitBox (  
    QRect hitBox)
```

Sets the hit box of the player.

Parameters

<i>hitBox</i>	The new hit box of the player.
---------------	--------------------------------

4.8.3.6 setItsPosition()

```
void Player::setItsPosition (  
    Position position)
```

Sets the position of the player.

Parameters

<i>position</i>	The new position of the player.
-----------------	---------------------------------

4.8.3.7 updatePos()

```
void Player::updatePos (  
    Direction direction)
```

Updates the position of the player based on the given direction.

Parameters

<i>direction</i>	The direction in which the player should move.
------------------	--

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

- **player.h**
- **player.cpp**

4.9 Position Struct Reference

Structure representing a position with x and y coordinates.

```
#include <typeDef.h>
```

Public Attributes

- int **posX**
- int **posY**

4.9.1 Detailed Description

Structure representing a position with x and y coordinates.

4.9.2 Member Data Documentation

4.9.2.1 posX

```
int Position::posX
```

The x-coordinate

4.9.2.2 posY

```
int Position::posY
```

The y-coordinate

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

- **typeDef.h**

4.10 PowerUp Class Reference

Represents a power-up in the game with a hitbox and type.

```
#include <powerup.h>
```

Public Member Functions

- **PowerUp** (**powerupType** type)
*Constructs a **PowerUp** (p. 34) object of a specified type.*
- **Position** **getItsPosition** ()
Gets the position of the power-up.
- **QRect** **getItsHitbox** ()
Gets the hitbox of the power-up.
- **powerupType** **getItsType** ()
Gets the type of the power-up.
- void **setItsPosition** (**Position** newPos)
Sets the position of the power-up.
- void **setItsHitbox** (**QRect** newHitbox)
Sets the hitbox of the power-up.

4.10.1 Detailed Description

Represents a power-up in the game with a hitbox and type.

This class manages the position, hitbox, and type of power-ups within the game.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 PowerUp()

```
PowerUp::PowerUp (
    powerupType type)
```

Constructs a **PowerUp** (p. 34) object of a specified type.

Parameters

<i>type</i>	The type of the power-up.
-------------	---------------------------

4.10.3 Member Function Documentation

4.10.3.1 getItsHitbox()

```
QRect PowerUp::getItsHitbox ()
```

Gets the hitbox of the power-up.

Returns

The QRect representing the power-up's hitbox.

4.10.3.2 getItsPosition()

```
Position PowerUp::getItsPosition ()
```

Gets the position of the power-up.

Returns

The **Position** (p. 33) struct representing the power-up's position.

4.10.3.3 getItsType()

```
powerupType PowerUp::getItsType ()
```

Gets the type of the power-up.

Returns

The powerupType enum representing the type of the power-up.

4.10.3.4 setItsHitbox()

```
void PowerUp::setItsHitbox (
    QRect newHitbox)
```

Sets the hitbox of the power-up.

Parameters

<i>newHitbox</i>	The new QRect representing the power-up's hitbox.
------------------	---

4.10.3.5 setItsPosition()

```
void PowerUp::setItsPosition (
    Position newPos)
```

Sets the position of the power-up.

Parameters

<i>newPos</i>	The new Position (p. 33) struct representing the power-up's position.
---------------	--

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

- **powerup.h**
- **powerup.cpp**

4.11 Spider Class Reference

Represents a spider in the game with a hitbox and movement capabilities.

```
#include <spider.h>
```

Public Member Functions

- **Spider** (int x, int y, int size)
*Constructs a **Spider** (p. 36) object with specified position and size.*
- QRect **getItsHitBox** ()
Gets the hitbox of the spider.
- void **setItsHitBox** (QRect hitbox)
Sets the hitbox of the spider.
- **Direction** **getItsDirection** ()
Gets the current direction of the spider.
- void **setItsDirection** (**Direction** direction)
Sets the direction of the spider.
- int **getItsHorizontaleDirection** ()
Gets the horizontal direction value of the spider.
- void **setItsHorizontaleDirection** (int horizontalDirection)
Sets the horizontal direction value of the spider.
- void **move** ()
Moves the spider based on its current direction and speed.

4.11.1 Detailed Description

Represents a spider in the game with a hitbox and movement capabilities.

This class manages the spider's position, hitbox, and movement behavior within the game.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Spider()

```
Spider::Spider (
    int x,
    int y,
    int size)
```

Constructs a **Spider** (p. 36) object with specified position and size.

Parameters

<i>x</i>	The x-coordinate of the spider's initial position.
<i>y</i>	The y-coordinate of the spider's initial position.
<i>size</i>	The size of the spider.

4.11.3 Member Function Documentation

4.11.3.1 getItsDirection()

```
Direction Spider::getItsDirection ()
```

Gets the current direction of the spider.

Returns

The **Direction** (p. 18) representing the spider's movement direction.

4.11.3.2 getItsHitBox()

```
QRect Spider::getItsHitBox ()
```

Gets the hitbox of the spider.

Returns

The QRect representing the spider's hitbox.

4.11.3.3 getItsHorizontaleDirection()

```
int Spider::getItsHorizontaleDirection ()
```

Gets the horizontal direction value of the spider.

Returns

An integer representing the horizontal direction.

4.11.3.4 move()

```
void Spider::move ()
```

Moves the spider based on its current direction and speed.

This method updates the spider's position according to its current direction and adjusts its movement speed.

4.11.3.5 setItsDirection()

```
void Spider::setItsDirection (  
    Direction direction)
```

Sets the direction of the spider.

Parameters

<i>direction</i>	The Direction (p. 18) representing the new movement direction of the spider.
------------------	---

4.11.3.6 setItsHitBox()

```
void Spider::setItsHitBox (  
    QRect hitbox)
```

Sets the hitbox of the spider.

Parameters

<i>hitbox</i>	The QRect representing the new hitbox of the spider.
---------------	---

4.11.3.7 setItsHorizontaleDirection()

```
void Spider::setItsHorizontaleDirection (  
    int horizontalDirection)
```

Sets the horizontal direction value of the spider.

Parameters

<i>horizontalDirection</i>	An integer representing the new horizontal direction.
----------------------------	---

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

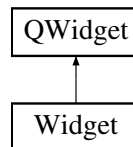
- **spider.h**
- spider.cpp

4.12 Widget Class Reference

Class representing the main game widget.

```
#include <widget.h>
```

Inheritance diagram for Widget:



Public Member Functions

- **Widget** (QWidget *parent=nullptr)
*Constructor for the **Widget** (p. 39) class.*
- **~Widget** ()
*Destructor for the **Widget** (p. 39) class.*

Protected Member Functions

- void **paintEvent** (QPaintEvent *event) override
*Overrides the paint event to draw the **Widget** (p. 39) components.*
- void **keyPressEvent** (QKeyEvent *event) override
Overrides the key press event to handle user input.
- void **keyReleaseEvent** (QKeyEvent *event) override
Overrides the key release event to handle user input.
- void **resizeEvent** (QResizeEvent *event) override
Overrides the resize event to handle widget resizing.
- void **drawCentipede** (QPainter &painter)
Draws the centipede on the widget using QPainter.
- void **drawPlayer** (QPainter &painter)
Draws the player on the widget using QPainter.
- void **drawBullet** (QPainter &painter)
Draws the bullet on the widget using QPainter.
- void **drawMushrooms** (QPainter &painter)
Draws the mushrooms on the widget using QPainter.

- void **drawPowerUps** (QPainter &painter)
Draws the powerups on the widget using QPainter.
- void **drawHeadUpDisplay** (QPainter &painter)
Draws the heads-up display on the widget using QPainter.
- void **drawSpider** (QPainter &painter)
Draws the spider display on the widget using QPainter.
- void **pauseGame** ()
Pauses the game and its timers.

4.12.1 Detailed Description

Class representing the main game widget.

This class manages the game mechanics and drawing on the widget. It handles player movement, centipede movement, bullet firing, powerups, and display updates.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Widget()

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

Constructor for the **Widget** (p. 39) class.

Parameters

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

4.12.3 Member Function Documentation

4.12.3.1 drawBullet()

```
void Widget::drawBullet (
    QPainter & painter) [protected]
```

Draws the bullet on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.2 drawCentipede()

```
void Widget::drawCentipede (
    QPainter & painter) [protected]
```

Draws the centipede on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.3 drawHeadUpDisplay()

```
void Widget::drawHeadUpDisplay (  
    QPainter & painter) [protected]
```

Draws the heads-up display on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.4 drawMushrooms()

```
void Widget::drawMushrooms (  
    QPainter & painter) [protected]
```

Draws the mushrooms on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.5 drawPlayer()

```
void Widget::drawPlayer (  
    QPainter & painter) [protected]
```

Draws the player on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.6 drawPowerUps()

```
void Widget::drawPowerUps (  
    QPainter & painter) [protected]
```

Draws the powerups on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.7 drawSpider()

```
void Widget::drawSpider (  
    QPainter & painter) [protected]
```

Draws the spider display on the widget using QPainter.

Parameters

<i>painter</i>	The QPainter object used for drawing.
----------------	---------------------------------------

4.12.3.8 keyPressEvent()

```
void Widget::keyPressEvent (  
    QKeyEvent * event) [override], [protected]
```

Overrides the key press event to handle user input.

Parameters

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

4.12.3.9 keyReleaseEvent()

```
void Widget::keyReleaseEvent (  
    QKeyEvent * event) [override], [protected]
```

Overrides the key release event to handle user input.

Parameters

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

4.12.3.10 paintEvent()

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

Overrides the paint event to draw the **Widget** (p. 39) components.

Parameters

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

4.12.3.11 pauseGame()

```
void Widget::pauseGame () [protected]
```

Pauses the game and its timers.

This method pauses all active timers and stops the game from updating.

4.12.3.12 resizeEvent()

```
void Widget::resizeEvent (  
    QResizeEvent * event) [override], [protected]
```

Overrides the resize event to handle widget resizing.

Parameters

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

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

- **widget.h**
- **widget.cpp**

Chapter 5

File Documentation

5.1 bodypart.h File Reference

Defines the **BodyPart** (p. 7) class, which represents a part of the centipede's body.

```
#include <QRect>
#include "typeDef.h"
```

Classes

- class **BodyPart**
Represents a part of the centipede's body.

5.1.1 Detailed Description

Defines the **BodyPart** (p. 7) class, which represents a part of the centipede's body.

5.2 bodypart.h

Go to the documentation of this file.

```
00001
00006 #ifndef BODYPART_H
00007 #define BODYPART_H
00008
00009 #include <QRect>
00010 #include "typeDef.h"
00011
00018 class BodyPart
00019 {
00020 private:
00021     BodyPart* itsParent = nullptr;
00022     BodyPart* itsChild = nullptr;
00023     QRect itsHitBox;
00024     Position itsPosition = {};
00025     Position itsTargetPos = {};
00026     Position itsPreviousPosition = {};
00028 public:
00033     BodyPart(int size);
00034
00038     ~BodyPart();
```

```

00039
00044     Position getItsPosition();
00045
00050     Position getItsPreviousPosition();
00051
00056     BodyPart* getItsChild();
00057
00062     BodyPart* getItsParent();
00063
00068     QRect getItsHitBox();
00069
00074     void setItsPosition(Position pos);
00075
00080     void setItsChild(BodyPart* child);
00081
00086     void setItsParent(BodyPart* parent);
00087
00092     void setItsHitBox(QRect hitbox);
00093
00098     void setItsTargetPos(Position targetPos);
00099
00103     void updatePos();
00104
00111     Position getNextTarget(Direction centipedeDir, int caseLength);
00112
00117     Position getItsTarget();
00118
00123     void addChild(BodyPart * child);
00124 };
00125
00126 #endif // BODYPART_H

```

5.3 bullet.h File Reference

Defines the **Bullet** (p. 11) class, which represents a bullet in the game.

```

#include <QRect>
#include "typeDef.h"

```

Classes

- class **Bullet**

Represents a bullet in the game.

5.3.1 Detailed Description

Defines the **Bullet** (p. 11) class, which represents a bullet in the game.

5.4 bullet.h

Go to the documentation of this file.

```

00001
00006 #ifndef BULLET_H
00007 #define BULLET_H
00008
00009 #include <QRect>
00010 #include "typeDef.h"
00011
00018 class Bullet
00019 {
00020 private:
00021     QRect itsHitBox;
00022     Position itsPosition;

```

```

00024 public:
00031     Bullet(int x, int y, int size);
00032
00036     ~Bullet();
00037
00043     void updatePos();
00044
00049     QRect getItsHitBox();
00050
00055     Position getItsPosition();
00056
00061     void setItsPosition(Position position);
00062
00067     void setItsHitBox(QRect hitbox);
00068 };
00069
00070 #endif // BULLET_H

```

5.5 centipede.h File Reference

Defines the **Centipede** (p. 13) class, which represents a centipede entity in a game.

```
#include "bodypart.h"
```

Classes

- class **Centipede**
Represents a centipede entity in a game.

5.5.1 Detailed Description

Defines the **Centipede** (p. 13) class, which represents a centipede entity in a game.

5.6 centipede.h

Go to the documentation of this file.

```

00001
00006 #ifndef CENTIPEDE_H
00007 #define CENTIPEDE_H
00008
00009 #include "bodypart.h"
00010
00018 class Centipede
00019 {
00020 private:
00021     BodyPart* itsHead;
00022     BodyPart* itsTail;
00023     Direction itsDirection;
00024     bool isGoingDown = false;
00025     bool reachedBottom = false;
00026     bool wasMovingRight = false;
00027     bool wasMovingLeft = false;
00029 public:
00034     Centipede(BodyPart* head);
00035
00039     ~Centipede();
00040
00045     BodyPart* getItsHead();
00046
00051     BodyPart* getItsTail();
00052
00057     void setItsTail(BodyPart* tail);
00058

```

```

00063     void setItsDirection(Direction dir);
00064
00070     Position getNextPosition(int distance);
00071
00076     Direction getItsDirection();
00077
00082     bool hasReachedBottom();
00083
00088     bool isVerticalDirection();
00089
00094     void setVerticalDirection(bool isCentipedeGoingDown);
00095
00100     void setHasReachedBottom(bool hasReachedBottom);
00101
00106     void setWasMovingRight(bool value);
00107
00112     void setWasMovingLeft(bool value);
00113
00118     bool getWasMovingRight();
00119
00124     bool getWasMovingLeft();
00125 };
00126
00127 #endif // CENTIPEDE_H

```

5.7 game.h File Reference

Defines the **Game** (p. 19) class, which represents the game management.

```

#include <vector>
#include <QRect>
#include "bullet.h"
#include "centipede.h"
#include "mushroom.h"
#include "player.h"
#include "typeDef.h"
#include "powerup.h"
#include "spider.h"

```

Classes

- class **Game**

Class representing the game management.

5.7.1 Detailed Description

Defines the **Game** (p. 19) class, which represents the game management.

5.8 game.h

Go to the documentation of this file.

```

00001
00006 #ifndef GAME_H
00007 #define GAME_H
00008
00009 #include <vector>
00010 #include <QRect>
00011 #include "bullet.h"
00012 #include "centipede.h"

```



```
00013 #include "mushroom.h"
00014 #include "player.h"
00015 #include "typeDef.h"
00016 #include "powerup.h"
00017 #include "spider.h"
00018
00025 class Game
00026 {
00027 private:
00028     int itsScore;
00029     std::vector<Centipede*>* itsCentipedes;
00030     std::vector<Mushroom*>* itsMushrooms;
00031     std::vector<PowerUp*> itsPowerups;
00032     std::vector<Bullet*> itsBullets;
00033     Player* itsPlayer;
00034     QRect itsBoard;
00035     QRect itsPlayerZone;
00036     QRect itsCentipedeZone;
00037     std::vector<Centipede*>* treatedCentipedes;
00038     int itsCurrentLevel = 1;
00039     bool isRafaleActive = false;
00040     bool isPiercingActive = false;
00041     Spider* itsSpider;
00042     std::vector<Mushroom*> itsMarkedMushroom;
00044 public:
00049     Game(QRect board);
00050
00054     ~Game();
00055
00059     void spawnCentipede();
00060
00064     void createMushrooms();
00065
00069     void shoot();
00070
00074     void moveBullets();
00075
00082     bool isColliding(Mushroom* mushroom, Player* player);
00083
00090     bool isColliding(Mushroom* mushroom, Bullet* bullet);
00091
00098     bool isColliding(Centipede* centipede, Bullet* bullet);
00099
00106     bool isColliding(Centipede* centipede, Mushroom* mushroom);
00107
00114     bool isColliding(QRect hitbox1, QRect hitbox2);
00115
00119     void checkCollisions();
00120
00126     void sliceCentipede(BodyPart* hitPart, Centipede* centipede);
00127
00132     void movePlayer(Direction &direction);
00133
00137     void movePowerUps();
00138
00143     std::vector<Centipede*>* getItsCentipedes();
00144
00149     std::vector<Mushroom*>* getItsMushrooms();
00150
00155     std::vector<Bullet*> getItsBullets();
00156
00161     Player* getItsPlayer();
00162
00167     int getItsScore();
00168
00173     QRect getItsBoard();
00174
00179     std::vector<PowerUp*> getItsPowerups();
00180
00185     int getCurrentLevel();
00186
00191     void setBoard(QRect board);
00192
00198     bool centipedeMushroomCollision(Centipede* centipede);
00199
00205     bool centipedeToCentipedeCollision(Centipede* centipede);
00206
00211     bool isLevelWon();
00212
00217     bool isGameLosed();
00218
00223     bool getIsRafaleActive();
00224
00229     bool getIsPiercingActive();
00230
00235     void setIsRafaleActive(bool isActive);
00236
```

```

00241     void setIsPiercingActive(bool isActive);
00242
00246     void moveCentipede();
00247
00254     bool centipedeBoardCollision(Centipede* centipede, QRect board);
00255
00259     void createSpider();
00260
00265     Spider* getItsSpider();
00266
00270     void moveSpider();
00271 };
00272
00273 #endif // GAME_H

```

5.9 leaderboard.h File Reference

Defines the **Leaderboard** (p. 27) class, which manages high scores in the game.

```

#include "typeDef.h"
#include <fstream>
#include <map>
#include <string>

```

Classes

- class **Leaderboard**
Manages and stores high scores in a leaderboard.

5.9.1 Detailed Description

Defines the **Leaderboard** (p. 27) class, which manages high scores in the game.

5.10 leaderboard.h

Go to the documentation of this file.

```

00001
00006 #ifndef LEADERBOARD_H
00007 #define LEADERBOARD_H
00008
00009 #include "typeDef.h"
00010 #include <fstream>
00011 #include <map>
00012 #include <string>
00013
00014 using namespace std;
00015
00020 class Leaderboard
00021 {
00022 public:
00027     Leaderboard(string filename);
00028
00033     map<string, int> getItsBestScores();
00034
00040     void addScore(int newScore, string username);
00041
00045     void extract();
00046
00050     void save();
00051
00052 private:
00053     string itsFileName;
00054     map<string, int> itsBestScores;
00055 };
00056
00057 #endif // LEADERBOARD_H

```

5.11 mushroom.h File Reference

Defines the **Mushroom** (p. 29) class, which represents a mushroom in the game.

```
#include <QRect>
#include "typeDef.h"
```

Classes

- class **Mushroom**

Class representing a mushroom in the game.

5.11.1 Detailed Description

Defines the **Mushroom** (p. 29) class, which represents a mushroom in the game.

5.12 mushroom.h

Go to the documentation of this file.

```
00001
00006 #ifndef MUSHROOM_H
00007 #define MUSHROOM_H
00008
00009 #include <QRect>
00010 #include "typeDef.h"
00011
00016 class Mushroom
00017 {
00018 private:
00019     int itsState;
00020     QRect itsHitBox;
00021     Position itsGridPosition;
00023 public:
00031     Mushroom(int x, int y, int size, Position gridPosition);
00032
00036     ~Mushroom();
00037
00041     void damage();
00042
00047     int getItsState();
00048
00053     QRect getItsHitBox();
00054
00059     Position getItsGridPosition();
00060
00065     void setItsHitBox(QRect hitBox);
00066
00071     void setItsGridPosition(Position position);
00072 };
00073
00074 #endif // MUSHROOM_H
```

5.13 player.h File Reference

Defines the **Player** (p. 31) class, which represents the player in the game.

```
#include <QRect>
#include "typeDef.h"
```

Classes

- class **Player**

Class representing the player in the game.

5.13.1 Detailed Description

Defines the **Player** (p. 31) class, which represents the player in the game.

5.14 player.h

Go to the documentation of this file.

```
00001
00006 #ifndef PLAYER_H
00007 #define PLAYER_H
00008
00009 #include <QRect>
00010 #include "typeDef.h"
00011
00016 class Player
00017 {
00018 private:
00019     int itsHP;
00020     QRect itsHitBox;
00021     Position itsPosition;
00023 public:
00029     Player(Position position, int size);
00030
00034     ~Player();
00035
00040     void updatePos(Direction direction);
00041
00045     void hit();
00046
00051     Position getItsPosition();
00052
00057     int getItsHp();
00058
00063     QRect getItsHitBox();
00064
00069     void setItsPosition(Position position);
00070
00075     void setItsHitBox(QRect hitBox);
00076
00081     void setItsHitBox(Position position);
00082 };
00083
00084 #endif // PLAYER_H
```

5.15 powerup.h File Reference

Header file for the **PowerUp** (p. 34) class.

```
#include <QRect>
#include "typeDef.h"
```

Classes

- class **PowerUp**

Represents a power-up in the game with a hitbox and type.

Enumerations

- enum **powerupType** { **rafale** , **transpercant** , **herbicide** }
Enumeration representing different types of power-ups.

5.15.1 Detailed Description

Header file for the **PowerUp** (p. 34) class.

5.15.2 Enumeration Type Documentation

5.15.2.1 powerupType

enum **powerupType**

Enumeration representing different types of power-ups.

Enumerator

rafale	'Rafale' power-up
transpercant	'Transpercant' power-up
herbicide	'Herbicide' power-up

5.16 powerup.h

Go to the documentation of this file.

```

00001
00006 #ifndef POWERUP_H
00007 #define POWERUP_H
00008
00009 #include <QRect>      // Include header file for QRect class
00010 #include "typeDef.h" // Include type definitions file
00011
00016 enum powerupType {
00017     rafale,
00018     transpercant,
00019     herbicide
00020 };
00021
00028 class PowerUp {
00029 private:
00030     powerupType itsType;
00031     QRect itsHitBox;
00032     Position itsPos;
00034 public:
00039     PowerUp(powerupType type);
00040
00045     Position getItsPosition();
00046
00051     QRect getItsHitbox();
00052
00057     powerupType getItsType();
00058
00063     void setItsPosition(Position newPos);
00064
00069     void setItsHitbox(QRect newHitbox);
00070 };
00071
00072 #endif // POWERUP_H

```

5.17 spider.h File Reference

Header file for the **Spider** (p. 36) class.

```
#include <QRect>
#include "typeDef.h"
```

Classes

- class **Spider**

Represents a spider in the game with a hitbox and movement capabilities.

5.17.1 Detailed Description

Header file for the **Spider** (p. 36) class.

5.18 spider.h

Go to the documentation of this file.

```
00001
00006 #include <QRect>          // Include header file for QRect class
00007 #include "typeDef.h"       // Include type definitions file
00008
00009 #ifndef SPIDER_H
00010 #define SPIDER_H
00011
00018 class Spider
00019 {
00020 private:
00021     QRect itsHitBox;
00022     Direction itsDirection;
00023     int itsHorizontaleDirection;
00025 public:
00032     Spider(int x, int y, int size);
00033
00038     QRect getItsHitBox();
00039
00044     void setItsHitBox(QRect hitbox);
00045
00050     Direction getItsDirection();
00051
00056     void setItsDirection(Direction direction);
00057
00062     int getItsHorizontaleDirection();
00063
00068     void setItsHorizontaleDirection(int horizontalDirection);
00069
00076     void move();
00077 };
00078
00079 #endif // SPIDER_H
```

5.19 typeDef.h File Reference

Defines common typedefs and constants used in the game.

```
#include <string>
```

Classes

- struct **Position**
Structure representing a position with x and y coordinates.
- struct **Direction**
Structure representing a direction with x and y components.

Variables

- const int **CENTIPEDE_LENGTH** = 8
- const int **CENTIPEDE_BODYPART_SIZE** = 13
- const int **CENTIPEDE_SPAWN_XPOS** = 15
- const int **CENTIPEDE_SPAWN_YPOS** = 0
- const int **PLAYER_SPEED** = 1
- const int **PLAYER_SIZE** = 5
- const int **MUSHROOM_SIZE** = 10
- const int **BULLET_SPEED** = 5
- const int **MUSHROOMS_AMOUNT** = 30
- const int **BOARD_WIDTH** = 30
- const int **BOARD_HEIGHT** = 31
- const int **POWERUP_DROP_RATE** = 30
- const int **POWERUP_RAFALE_DURATION** = 4
- const int **POWERUP_RAFALE_FIRERATE** = 4
- const int **POWERUP_PIERCING_DURATION** = 5
- const bool **SHOW_HITBOXES** = false
- const int **INCREMENT_INTERVAL** = 5
- const int **SPIDER_SPEED** = 1
- const std::string **SAVEFILE_NAME** = "leaderboard.txt"

5.19.1 Detailed Description

Defines common typedefs and constants used in the game.

5.19.2 Variable Documentation

5.19.2.1 BOARD_HEIGHT

```
const int BOARD_HEIGHT = 31
```

The height of the game board

5.19.2.2 BOARD_WIDTH

```
const int BOARD_WIDTH = 30
```

The width of the game board

5.19.2.3 BULLET_SPEED

```
const int BULLET_SPEED = 5
```

The speed of the bullet

5.19.2.4 CENTIPEDE_BODYPART_SIZE

```
const int CENTIPEDE_BODYPART_SIZE = 13
```

The size of each body part of the centipede

5.19.2.5 CENTIPEDE_LENGTH

```
const int CENTIPEDE_LENGTH = 8
```

The length of the centipede

5.19.2.6 CENTIPEDE_SPAWN_XPOS

```
const int CENTIPEDE_SPAWN_XPOS = 15
```

The initial x-coordinate of the centipede

5.19.2.7 CENTIPEDE_SPAWN_YPOS

```
const int CENTIPEDE_SPAWN_YPOS = 0
```

The initial y-coordinate of the centipede

5.19.2.8 INCREMENT_INTERVAL

```
const int INCREMENT_INTERVAL = 5
```

The interval for incrementing the spider's movement

5.19.2.9 MUSHROOM_SIZE

```
const int MUSHROOM_SIZE = 10
```

The size of the mushroom

5.19.2.10 MUSHROOMS_AMOUNT

```
const int MUSHROOMS_AMOUNT = 30
```

The number of mushrooms in the game

5.19.2.11 PLAYER_SIZE

```
const int PLAYER_SIZE = 5
```

The size of the player

5.19.2.12 PLAYER_SPEED

```
const int PLAYER_SPEED = 1
```

The speed of the player

5.19.2.13 POWERUP_DROPRATE

```
const int POWERUP_DROPRATE = 30
```

The chance in percent for a powerup to appear when a mushroom is broken by a bullet

5.19.2.14 POWERUP_PIERCING_DURATION

```
const int POWERUP_PIERCING_DURATION = 5
```

The duration in second of the 'piercing' powerup

5.19.2.15 POWERUP_RAFALE_DURATION

```
const int POWERUP_RAFALE_DURATION = 4
```

The duration in seconds of the 'rafale' powerup

5.19.2.16 POWERUP_RAFALE_FIRERATE

```
const int POWERUP_RAFALE_FIRERATE = 4
```

The number of shots per second of the 'rafale' powerup

5.19.2.17 SAVEFILE_NAME

```
const std::string SAVEFILE_NAME = "leaderboard.txt"
```

The name of the file to save leaderboard data

5.19.2.18 SHOW_HITBOXES

```
const bool SHOW_HITBOXES = false
```

Flag indicating whether to show hitboxes

5.19.2.19 SPIDER_SPEED

```
const int SPIDER_SPEED = 1
```

The speed of the spider

5.20 typeDef.h

Go to the documentation of this file.

```
00001
00006 #ifndef TYPEDEF_H
00007 #define TYPEDEF_H
00008
00009 #include <string>
00010
00015 struct Position
00016 {
00017     int posX;
00018     int posY;
00019 };
00020
00025 struct Direction
00026 {
00027     int dirX;
00028     int dirY;
00029 };
00030
00031 // Variables for centipede
00032 const int CENTIPEDE_LENGTH = 8;
00033 const int CENTIPEDE_BODYPART_SIZE = 13;
00034 const int CENTIPEDE_SPAWN_XPOS = 15;
00035 const int CENTIPEDE_SPAWN_YPOS = 0;
00037 // Variables for player
00038 const int PLAYER_SPEED = 1;
00039 const int PLAYER_SIZE = 5;
00041 // Variables for mushroom
00042 const int MUSHROOM_SIZE = 10;
00044 // Variables for bullet
00045 const int BULLET_SPEED = 5;
00047 // Variables for game
00048 const int MUSHROOMS_AMOUNT = 30;
00049 const int BOARD_WIDTH = 30;
00050 const int BOARD_HEIGHT = 31;
00052 // Variables for powerups
00053 const int POWERUP_DROPRATE = 30;
00054 const int POWERUP_RAFALE_DURATION = 4;
00055 const int POWERUP_RAFALE_FIRERATE = 4;
00056 const int POWERUP_PIERCING_DURATION = 5;
00058 const bool SHOW_HITBOXES = false;
00060 // Variables for the Spider
00061 const int INCREMENT_INTERVAL = 5;
00062 const int SPIDER_SPEED = 1;
00064 const std::string SAVEFILE_NAME = "leaderboard.txt";
00066 #endif // TYPEDEF_H
```

5.21 widget.h File Reference

Defines the **Widget** (p. 39) class, which represents the main game widget.

```
#include <QWidget>
#include <QTimer>
#include <QImage>
#include <QPainter>
#include <QKeyEvent>
#include <QFontMetrics>
#include <QResizeEvent>
#include "game.h"
#include "typeDef.h"
#include "leaderboard.h"
```

Classes

- class **Widget**

Class representing the main game widget.

5.21.1 Detailed Description

Defines the **Widget** (p. 39) class, which represents the main game widget.

5.22 widget.h

Go to the documentation of this file.

```
00001
00006 #ifndef WIDGET_H
00007 #define WIDGET_H
00008
00009 #include <QWidget>
00010 #include <QTimer>
00011 #include <QImage>
00012 #include <QPainter>
00013 #include <QKeyEvent>
00014 #include <QFontMetrics>
00015 #include <QResizeEvent>
00016 #include "game.h"
00017 #include "typeDef.h"
00018 #include "leaderboard.h"
00019
00020 QT_BEGIN_NAMESPACE
00021 namespace Ui {
00022 class Widget;
00023 }
00024 QT_END_NAMESPACE
00025
00033 class Widget : public QWidget
00034 {
00035     Q_OBJECT
00036
00037 public:
00042     Widget(QWidget *parent = nullptr);
00043
00047     ~Widget();
00048
00049 protected:
00054     void paintEvent(QPaintEvent *event) override;
00055
00060     void keyPressEvent(QKeyEvent * event) override;
00061
00066     void keyReleaseEvent(QKeyEvent * event) override;
00067
00072     void resizeEvent(QResizeEvent *event) override;
00073
00078     void drawCentipede(QPainter & painter);
00079
00084     void drawPlayer(QPainter & painter);
00085
00090     void drawBullet(QPainter & painter);
00091
00096     void drawMushrooms(QPainter & painter);
00097
00102     void drawPowerUps(QPainter & painter);
00103
00108     void drawHeadUpDisplay(QPainter & painter);
00109
00114     void drawSpider(QPainter & painter);
00115
00121     void pauseGame();
00122
00123 private slots:
00129     void movePlayer();
00130
00136     void moveCentipede();
00137
00143     void moveBullet();
00144
```

```
00150     void movePowerUps();
00151
00160     void startGame(int level = 1);
00161
00167     void resumeGame();
00168
00175     void endGame();
00176
00182     void backToMenu();
00183
00189     void displayLeaderboard();
00190
00196     void processNewScore();
00197
00203     void goToHowToPlay();
00204
00210     void rafaleShot();
00211
00217     void piercingEnd();
00218
00224     void moveSpider();
00225
00231     void spiderAppear();
00232
00233 private:
00234     Ui::Widget *ui;
00235     QTimer * itsDisplayTimer = nullptr;
00236     QTimer * itsCentipedeTimer = nullptr;
00237     QTimer * itsBulletTimer = nullptr;
00238     QTimer * itsPlayerTimer = nullptr;
00239     QTimer * itsPowerUpMovementTimer = nullptr;
00240     QTimer * itsRafaleTimer = nullptr;
00241     QTimer * itsPiercingTimer = nullptr;
00242     QTimer * itsSpiderAppearTimer = nullptr;
00243     QTimer * itsSpiderTimer = nullptr;
00244     Game * itsGame = nullptr;
00245     Leaderboard * itsLeaderboard = nullptr;
00246     QImage itsCentiHeadImg;
00247     QImage itsCentiBodyImg;
00248     QImage itsCentiTailImg;
00249     QImage itsPlayerImg;
00250     QImage itsMushState1Img;
00251     QImage itsMushState2Img;
00252     QImage itsMushState3Img;
00253     QImage itsMushState4Img;
00254     QImage itsSpiderImg;
00255     QImage itsBulletImg;
00256     QImage itsRafalePuiImg;
00257     QImage itsTranspercantPuiImg;
00258     QImage itsHerbicidePuiImg;
00259     QImage itsGrassTexture;
00260     QImage itsDarkGrassTexture;
00261     Direction itsPlayerDirection;
00262     bool isGameStarted = false;
00263     bool isGamePaused = false;
00264     int remainingRafaleShots;
00265     int itsElapsedTime;
00266     int itsSpiderAppearProbability;
00267 };
00268
00269 #endif // WIDGET_H
```

Index

- addChild
 - BodyPart, 8
- addScore
 - Leaderboard, 28
- BOARD_HEIGHT
 - typeDef.h, 55
- BOARD_WIDTH
 - typeDef.h, 55
- BodyPart, 7
 - addChild, 8
 - BodyPart, 8
 - getItsChild, 8
 - getItsHitBox, 8
 - getItsParent, 9
 - getItsPosition, 9
 - getItsPreviousPosition, 9
 - getItsTarget, 9
 - getNextTarget, 9
 - setItsChild, 10
 - setItsHitBox, 10
 - setItsParent, 10
 - setItsPosition, 10
 - setItsTargetPos, 11
- bodypart.h, 45
- Bullet, 11
 - Bullet, 12
 - getItsHitBox, 12
 - getItsPosition, 12
 - setItsHitBox, 12
 - setItsPosition, 13
 - updatePos, 13
- bullet.h, 46
- BULLET_SPEED
 - typeDef.h, 55
- Centipede, 13
 - Centipede, 14
 - getItsDirection, 15
 - getItsHead, 15
 - getItsTail, 15
 - getNextPosition, 15
 - getWasMovingLeft, 15
 - getWasMovingRight, 16
 - hasReachedBottom, 16
 - isVerticalDirection, 16
 - setHasReachedBottom, 16
 - setItsDirection, 17
 - setItsTail, 17
 - setVerticalDirection, 17
 - setWasMovingLeft, 17
 - setWasMovingRight, 18
- centipede.h, 47
- CENTPEDE_BODYPART_SIZE
 - typeDef.h, 56
- CENTPEDE_LENGTH
 - typeDef.h, 56
- CENTPEDE_SPAWN_XPOS
 - typeDef.h, 56
- CENTPEDE_SPAWN_YPOS
 - typeDef.h, 56
- centipedeBoardCollision
 - Game, 21
- centipedeMushroomCollision
 - Game, 21
- centipedeToCentipedeCollision
 - Game, 21
- Direction, 18
 - dirX, 18
 - dirY, 18
- dirX
 - Direction, 18
- dirY
 - Direction, 18
- drawBullet
 - Widget, 40
- drawCentipede
 - Widget, 40
- drawHeadUpDisplay
 - Widget, 41
- drawMushrooms
 - Widget, 41
- drawPlayer
 - Widget, 41
- drawPowerUps
 - Widget, 41
- drawSpider
 - Widget, 42
- Game, 19
 - centipedeBoardCollision, 21
 - centipedeMushroomCollision, 21
 - centipedeToCentipedeCollision, 21
 - Game, 21
 - getCurrentLevel, 22
 - getIsPiercingActive, 22
 - getIsRafaleActive, 22
 - getItsBoard, 22
 - getItsBullets, 22

- getItsCentipedes, 23
- getItsMushrooms, 23
- getItsPlayer, 23
- getItsPowerups, 23
- getItsScore, 23
- getItsSpider, 24
- isColliding, 24, 25
- isGameLost, 25
- isLevelWon, 26
- movePlayer, 26
- setBoard, 26
- setIsPiercingActive, 26
- setIsRafaleActive, 27
- sliceCentipede, 27
- game.h, 48
- getCurrentLevel
 - Game, 22
- getIsPiercingActive
 - Game, 22
- getIsRafaleActive
 - Game, 22
- getItsBestScores
 - Leaderboard, 28
- getItsBoard
 - Game, 22
- getItsBullets
 - Game, 22
- getItsCentipedes
 - Game, 23
- getItsChild
 - BodyPart, 8
- getItsDirection
 - Centipede, 15
 - Spider, 37
- getItsGridPosition
 - Mushroom, 30
- getItsHead
 - Centipede, 15
- getItsHitBox
 - BodyPart, 8
 - Bullet, 12
 - Mushroom, 30
 - Player, 32
 - Spider, 37
- getItsHitbox
 - PowerUp, 35
- getItsHorizontaleDirection
 - Spider, 37
- getItsHp
 - Player, 32
- getItsMushrooms
 - Game, 23
- getItsParent
 - BodyPart, 9
- getItsPlayer
 - Game, 23
- getItsPosition
 - BodyPart, 9
- Bullet, 12
- Player, 32
- PowerUp, 35
- getItsPowerups
 - Game, 23
- getItsPreviousPosition
 - BodyPart, 9
- getItsScore
 - Game, 23
- getItsSpider
 - Game, 24
- getItsState
 - Mushroom, 30
- getItsTail
 - Centipede, 15
- getItsTarget
 - BodyPart, 9
- getItsType
 - PowerUp, 35
- getNextPosition
 - Centipede, 15
- getNextTarget
 - BodyPart, 9
- getWasMovingLeft
 - Centipede, 15
- getWasMovingRight
 - Centipede, 16
- hasReachedBottom
 - Centipede, 16
- herbicide
 - powerup.h, 53
- INCREMENT_INTERVAL
 - typeDef.h, 56
- isColliding
 - Game, 24, 25
- isGameLost
 - Game, 25
- isLevelWon
 - Game, 26
- isVerticalDirection
 - Centipede, 16
- keyPressEvent
 - Widget, 42
- keyReleaseEvent
 - Widget, 42
- Leaderboard, 27
 - addScore, 28
 - getItsBestScores, 28
 - Leaderboard, 28
- leaderboard.h, 50
- move
 - Spider, 38
- movePlayer
 - Game, 26

Mushroom, 29
 getItsGridPosition, 30
 getItsHitBox, 30
 getItsState, 30
 Mushroom, 29
 setItsGridPosition, 30
 setItsHitBox, 30
mushroom.h, 51
MUSHROOM_SIZE
 typeDef.h, 56
MUSHROOMS_AMOUNT
 typeDef.h, 56

paintEvent
 Widget, 42
pauseGame
 Widget, 43
Player, 31
 getItsHitBox, 32
 getItsHp, 32
 getItsPosition, 32
 Player, 31
 setItsHitBox, 32, 33
 setItsPosition, 33
 updatePos, 33
player.h, 51
PLAYER_SIZE
 typeDef.h, 56
PLAYER_SPEED
 typeDef.h, 57
Position, 33
 posX, 34
 posY, 34
posX
 Position, 34
posY
 Position, 34
PowerUp, 34
 getItsHitbox, 35
 getItsPosition, 35
 getItsType, 35
 PowerUp, 35
 setItsHitbox, 35
 setItsPosition, 36
powerup.h, 52
 herbicide, 53
 powerupType, 53
 rafale, 53
 transpercant, 53
POWERUP_DROPRATE
 typeDef.h, 57
POWERUP_PIERCING_DURATION
 typeDef.h, 57
POWERUP_RAFALE_DURATION
 typeDef.h, 57
POWERUP_RAFALE_FIRERATE
 typeDef.h, 57
powerupType
 powerup.h, 53

rafale
 powerup.h, 53
resizeEvent
 Widget, 43

SAVEFILE_NAME
 typeDef.h, 57
setBoard
 Game, 26
setHasReachedBottom
 Centipede, 16
setIsPiercingActive
 Game, 26
setIsRafaleActive
 Game, 27
setItsChild
 BodyPart, 10
setItsDirection
 Centipede, 17
 Spider, 38
setItsGridPosition
 Mushroom, 30
setItsHitBox
 BodyPart, 10
 Bullet, 12
 Mushroom, 30
 Player, 32, 33
 Spider, 38
setItsHitbox
 PowerUp, 35
setItsHorizontaleDirection
 Spider, 38
setItsParent
 BodyPart, 10
setItsPosition
 BodyPart, 10
 Bullet, 13
 Player, 33
 PowerUp, 36
setItsTail
 Centipede, 17
setItsTargetPos
 BodyPart, 11
setVerticalDirection
 Centipede, 17
setWasMovingLeft
 Centipede, 17
setWasMovingRight
 Centipede, 18
SHOW_HITBOXES
 typeDef.h, 57
sliceCentipede
 Game, 27
Spider, 36
 getItsDirection, 37
 getItsHitBox, 37
 getItsHorizontaleDirection, 37
 move, 38
 setItsDirection, 38

- setItsHitBox, 38
- setItsHorizontaleDirection, 38
- Spider, 37
- spider.h, 54
- SPIDER_SPEED
 - typeDef.h, 57
- transpercant
 - powerup.h, 53
- typeDef.h, 54
 - BOARD_HEIGHT, 55
 - BOARD_WIDTH, 55
 - BULLET_SPEED, 55
 - CENTIPEDE_BODYPART_SIZE, 56
 - CENTIPEDE_LENGTH, 56
 - CENTIPEDE_SPAWN_XPOS, 56
 - CENTIPEDE_SPAWN_YPOS, 56
 - INCREMENT_INTERVAL, 56
 - MUSHROOM_SIZE, 56
 - MUSHROOMS_AMOUNT, 56
 - PLAYER_SIZE, 56
 - PLAYER_SPEED, 57
 - POWERUP_DROP_RATE, 57
 - POWERUP_PIERCING_DURATION, 57
 - POWERUP_RAFALE_DURATION, 57
 - POWERUP_RAFALE_FIRERATE, 57
 - SAVEFILE_NAME, 57
 - SHOW_HITBOXES, 57
 - SPIDER_SPEED, 57
- updatePos
 - Bullet, 13
 - Player, 33
- Widget, 39
 - drawBullet, 40
 - drawCentipede, 40
 - drawHeadUpDisplay, 41
 - drawMushrooms, 41
 - drawPlayer, 41
 - drawPowerUps, 41
 - drawSpider, 42
 - keyPressEvent, 42
 - keyReleaseEvent, 42
 - paintEvent, 42
 - pauseGame, 43
 - resizeEvent, 43
 - Widget, 40
- widget.h, 58