RetroG24_C22_Centipede

v.3

Generated by Doxygen 1.11.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

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

size	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

child	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

centipedeDir	The direction of movement for the centipede.
caseLength	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

child	Pointer to the child body part.

4.1.3.10 setItsHitBox()

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

Sets the hitbox of the body part.

Parameters

hitbox	The new hitbox of the body part.

4.1.3.11 setItsParent()

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

Sets the parent body part.

Parameters

parent	Pointer to the parent body part.

4.1.3.12 setItsPosition()

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

Sets the position of the body part.

Parameters

pos	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

targetPos	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

x	The x-coordinate of the bullet's initial position.
y	The y-coordinate of the bullet's initial position.
size	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

hitbox	The new hitbox of the bullet.

4.2.3.4 setItsPosition()

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

Sets the position of the bullet.

Parameters

position	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

head	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

distance	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

hasReachedBottom	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

dir	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

tail	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

isCentipedeGoingDown	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

value	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

value	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 **getIsRafaleActive** ()

  Checks if the 'rafale' powerup is active.
- bool **getIsPiercingActive** ()

  Checks if the 'transpercant' powerup is active.
- void **setIsRafaleActive** (bool isActive)

  Sets the state of the 'rafale' powerup.
- void **setIsPiercingActive** (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

board	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

centipede	Pointer to the centipede.
board	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

centipede	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

centipede	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

centipede	Pointer to the centipede.
bullet	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

centipede	Pointer to the centipede.
mushroom	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

mushroom	Pointer to the mushroom.
bullet	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

mushroom	Pointer to the mushroom.
player	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

hitbox1	The first rectangle.
hitbox2	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

direction	Reference to the direction of movement.

4.5.3.23 setBoard()

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

Sets the game board rectangle.

Parameters

board	The new game board rectangle.

4.5.3.24 setIsPiercingActive()

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

Sets the state of the 'transpercant' powerup.

Parameters

isActive	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

isActive	The state to set for the 'rafale' powerup.

4.5.3.26 sliceCentipede()

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

Slices a centipede when hit.

Parameters

hittedPart	Pointer to the hit body part of the centipede.
centipede	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

filename	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

newScore	The new score to add.
username	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

x	The x-coordinate of the mushroom.
y	The y-coordinate of the mushroom.
size	The size of the mushroom.
gridPosition	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

position	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

hitBox	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

position	The initial position of the player.
size	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

position	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

hitBox	The new hit box of the player.

4.8.3.6 setItsPosition()

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

Sets the position of the player.

Parameters

position	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

direction	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

type	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

newHitbox	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

newPos	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

x	The x-coordinate of the spider's initial position.
y	The y-coordinate of the spider's initial position.
size	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

direction	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

hitbox	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

horizontalDirection	An integer representing the new horizontal direction.

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

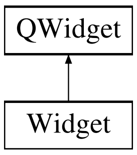
- **spider.h**
- spider.cpp

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

parent	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

painter	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

painter	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

painter	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

painter	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

painter	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

painter	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

painter	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

event	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

event	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

event	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

event	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.

```

#ifndef BODYPART_H
#define BODYPART_H

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

class BodyPart
{
private:
    BodyPart* itsParent = nullptr;
    BodyPart* itsChild = nullptr;
    QRect itsHitBox;
    Position itsPosition = {};
    Position itsTargetPos = {};
    Position itsPreviousPosition = {};
public:
    BodyPart(int size);

    ~BodyPart();
    Position getItsPosition();

    Position getItsPreviousPosition();

    BodyPart* getItsChild();

    BodyPart* getItsParent();

    QRect getItsHitBox();

    void setItsPosition(Position pos);

    void setItsChild(BodyPart* child);

    void setItsParent(BodyPart* parent);

    void setItsHitBox(QRect hitbox);

    void setItsTargetPos(Position targetPos);

    void updatePos();

    Position getNextTarget(Direction centipedeDir, int caseLength);

    Position getItsTarget();

    void addChild(BodyPart * child);
};

#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.

```

#ifndef BULLET_H
#define BULLET_H

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

class Bullet
{
private:
    QRect itsHitBox;
    Position itsPosition;
public:
    Bullet(int x, int y, int size);

    ~Bullet();

    void updatePos();

    QRect getItsHitBox();

    Position getItsPosition();

    void setItsPosition(Position position);

    void setItsHitBox(QRect hitbox);
};

#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.

```

#ifndef CENTIPEDE_H
#define CENTIPEDE_H

#include "bodypart.h"

class Centipede
{
private:
    BodyPart* itsHead;
    BodyPart* itsTail;
    Direction itsDirection;
    bool isGoingDown = false;
    bool reachedBottom = false;
    bool wasMovingRight = false;
    bool wasMovingLeft = false;
public:
    Centipede(BodyPart* head);

    ~Centipede();

    BodyPart* getItsHead();

    BodyPart* getItsTail();

    void setItsTail(BodyPart* tail);

    void setItsDirection(Direction dir);

    Position getNextPosition(int distance);

    Direction getItsDirection();

    bool hasReachedBottom();

    bool isVerticalDirection();

    void setVerticalDirection(bool isCentipedeGoingDown);

    void setHasReachedBottom(bool hasReachedBottom);

    void setWasMovingRight(bool value);

    void setWasMovingLeft(bool value);

    bool getWasMovingRight();

    bool getWasMovingLeft();
};

#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.

```

#ifndef GAME_H
#define GAME_H

#include <vector>
#include <QRect>
#include "bullet.h"
#include "centipede.h"
```

```cpp
#include "mushroom.h"
#include "player.h"
#include "typeDef.h"
#include "powerup.h"
#include "spider.h"

class Game
{
private:
    int itsScore;
    std::vector<Centipede*>* itsCentipedes;
    std::vector<Mushroom*>* itsMushrooms;
    std::vector<PowerUp*> itsPowerups;
    std::vector<Bullet*> itsBullets;
    Player* itsPlayer;
    QRect itsBoard;
    QRect itsPlayerZone;
    QRect itsCentipedeZone;
    std::vector<Centipede*>* treatedCentipedes;
    int itsCurrentLevel = 1;
    bool isRafaleActive = false;
    bool isPiercingActive = false;
    Spider* itsSpider;
    std::vector<Mushroom*> itsMarkedMushroom;
public:
    Game(QRect board);

    ~Game();

    void spawnCentipede();

    void createMushrooms();

    void shoot();

    void moveBullets();

    bool isColliding(Mushroom* mushroom, Player* player);

    bool isColliding(Mushroom* mushroom, Bullet* bullet);

    bool isColliding(Centipede* centipede, Bullet* bullet);

    bool isColliding(Centipede* centipede, Mushroom* mushroom);

    bool isColliding(QRect hitbox1, QRect hitbox2);

    void checkCollisions();

    void sliceCentipede(BodyPart* hittedPart, Centipede* centipede);

    void movePlayer(Direction &direction);

    void movePowerUps();

    std::vector<Centipede*>* getItsCentipedes();

    std::vector<Mushroom*>* getItsMushrooms();

    std::vector<Bullet*> getItsBullets();

    Player* getItsPlayer();

    int getItsScore();

    QRect getItsBoard();

    std::vector<PowerUp*> getItsPowerups();

    int getCurrentLevel();

    void setBoard(QRect board);

    bool centipedeMushroomCollision(Centipede* centipede);

    bool centipedeToCentipedeCollision(Centipede* centipede);

    bool isLevelWon();

    bool isGameLosed();

    bool getIsRafaleActive();

    bool getIsPiercingActive();

    void setIsRafaleActive(bool isActive);

    void setIsPiercingActive(bool isActive);

    void moveCentipede();

    bool centipedeBoardCollision(Centipede* centipede, QRect board);

    void createSpider();

    Spider* getItsSpider();

    void moveSpider();
};

#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.

```

#ifndef LEADERBOARD_H
#define LEADERBOARD_H

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

using namespace std;

class Leaderboard
{
public:
    Leaderboard(string filename);

    map<string, int> getItsBestScores();

    void addScore(int newScore, string username);

    void extract();

    void save();

private:
    string itsFileName;
    map<string, int> itsBestScores;
};

#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.

```

#ifndef MUSHROOM_H
#define MUSHROOM_H

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

class Mushroom
{
private:
    int itsState;
    QRect itsHitBox;
    Position itsGridPosition;
public:
    Mushroom(int x, int y, int size, Position gridPosition);

    ~Mushroom();

    void damage();

    int getItsState();

    QRect getItsHitBox();

    Position getItsGridPosition();

    void setItsHitBox(QRect hitBox);

    void setItsGridPosition(Position position);
};

#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.

```

#ifndef PLAYER_H
#define PLAYER_H

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

class Player
{
private:
    int itsHP;
    QRect itsHitBox;
    Position itsPosition;
public:
    Player(Position position, int size);

    ~Player();

    void updatePos(Direction direction);

    void hit();

    Position getItsPosition();

    int getItsHp();

    QRect getItsHitBox();

    void setItsPosition(Position position);

    void setItsHitBox(QRect hitBox);

    void setItsHitBox(Position position);
};

#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.

```
#ifndef POWERUP_H
#define POWERUP_H

#include <QRect>    // Include header file for QRect class
#include "typeDef.h" // Include type definitions file

enum powerupType {
    rafale,
    transpercant,
    herbicide
};

class PowerUp {
private:
    powerupType itsType;
    QRect itsHitBox;
    Position itsPos;
public:
    PowerUp(powerupType type);

    Position getItsPosition();

    QRect getItsHitbox();

    powerupType getItsType();

    void setItsPosition(Position newPos);

    void setItsHitbox(QRect newHitbox);
};

#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.

```

#include <QRect>      // Include header file for QRect class
#include "typeDef.h" // Include type definitions file

#ifndef SPIDER_H
#define SPIDER_H

class Spider
{
private:
    QRect itsHitBox;
    Direction itsDirection;
    int itsHorizontaleDirection;
public:
    Spider(int x, int y, int size);

    QRect getItsHitBox();

    void setItsHitBox(QRect hitbox);

    Direction getItsDirection();

    void setItsDirection(Direction direction);

    int getItsHorizontaleDirection();

    void setItsHorizontaleDirection(int horizontalDirection);

    void move();
};

#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_DROPRATE** = 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.

```

#ifndef TYPEDEF_H
#define TYPEDEF_H

#include <string>

struct Position
{
    int posX;
    int posY;
};

struct Direction
{
    int dirX;
    int dirY;
};

// Variables for centipede
const int CENTIPEDE_LENGTH = 8;
const int CENTIPEDE_BODYPART_SIZE = 13;
const int CENTIPEDE_SPAWN_XPOS = 15;
const int CENTIPEDE_SPAWN_YPOS = 0;
// Variables for player
const int PLAYER_SPEED = 1;
const int PLAYER_SIZE = 5;
// Variables for mushroom
const int MUSHROOM_SIZE = 10;
// Variables for bullet
const int BULLET_SPEED = 5;
// Variables for game
const int MUSHROOMS_AMOUNT = 30;
const int BOARD_WIDTH = 30;
const int BOARD_HEIGHT = 31;
// Variables for powerups
const int POWERUP_DROPRATE = 30;
const int POWERUP_RAFALE_DURATION = 4;
const int POWERUP_RAFALE_FIRERATE = 4;
const int POWERUP_PIERCING_DURATION = 5;
const bool SHOW_HITBOXES = false;
// Variables for the Spider
const int INCREMENT_INTERVAL = 5;
const int SPIDER_SPEED = 1;
const std::string SAVEFILE_NAME = "leaderboard.txt";
#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.

```

#ifndef WIDGET_H
#define WIDGET_H

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

QT_BEGIN_NAMESPACE
namespace Ui {
class Widget;
}
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT

public:
    Widget(QWidget *parent = nullptr);

    ~Widget();

protected:
    void paintEvent(QPaintEvent *event) override;

    void keyPressEvent(QKeyEvent * event) override;

    void keyReleaseEvent(QKeyEvent * event) override;

    void resizeEvent(QResizeEvent *event) override;

    void drawCentipede(QPainter & painter);

    void drawPlayer(QPainter & painter);

    void drawBullet(QPainter & painter);

    void drawMushrooms(QPainter & painter);

    void drawPowerUps(QPainter & painter);

    void drawHeadUpDisplay(QPainter & painter);

    void drawSpider(QPainter & painter);

    void pauseGame();

private slots:
    void movePlayer();

    void moveCentipede();

    void moveBullet();

    void movePowerUps();

    void startGame(int level = 1);

    void resumeGame();

    void endGame();

    void backToMenu();

    void displayLeaderboard();

    void processNewScore();

    void goToHowToPlay();

    void rafaleShot();

    void piercingEnd();

    void moveSpider();

    void spiderAppear();

private:
    Ui::Widget *ui;
    QTimer * itsDisplayTimer = nullptr;
    QTimer * itsCentipedeTimer = nullptr;
    QTimer * itsBulletTimer = nullptr;
    QTimer * itsPlayerTimer = nullptr;
    QTimer * itsPowerUpMovementTimer = nullptr;
    QTimer * itsRafaleTimer = nullptr;
    QTimer * itsPiercingTimer = nullptr;
    QTimer * itsSpiderAppearTimer = nullptr;
    QTimer * itsSpiderTimer = nullptr;
    Game * itsGame = nullptr;
    Leaderboard * itsLeaderboard = nullptr;
    QImage itsCentiHeadImg;
    QImage itsCentiBodyImg;
    QImage itsCentiTailImg;
    QImage itsPlayerImg;
    QImage itsMushState1Img;
    QImage itsMushState2Img;
    QImage itsMushState3Img;
    QImage itsMushState4Img;
    QImage itsSpiderImg;
    QImage itsBulletImg;
    QImage itsRafalePuImg;
    QImage itsTranspercantPuImg;
    QImage itsHerbicidePuImg;
    QImage itsGrassTexture;
    QImage itsDarkGrassTexture;
    Direction itsPlayerDirection;
    bool isGameStarted = false;
    bool isGamePaused = false;
    int remainingRafaleShots;
    int itsElapsedTime;
    int itsSpiderAppearProbability;
};

#endif // WIDGET_H
```

Index