Snake

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Food Class Reference

The food that can be eaten by the snake.

`#include <Food.h>`

Public Member Functions

- Food (const sf::Vector2u winSize, sf::Texture *texture)
  Constructor which creates a piece of food.
- void reset ()
  Resets the foods position to a random position.
- void draw (sf::RenderWindow &window)
  Draws the piece of food.
- sf::Vector2f getPosition ()
  Returns the foods position.
- ~Food ()
  Destructor that destroys a piece of food.

2.1.1 Detailed Description

The food that can be eaten by the snake.

This class displays the food which can be eaten by the players snake. The food is represented by a sf::Rectangle↩ Shape.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Food::Food (const sf::Vector2u *winSize*, sf::Texture * *texture*)

Constructor which creates a piece of food.

Parameters

in	*winSize*	The size of the games window.
in	*texture*	The foods texture.

This method creates a piece of food. It sets the texture, size and origin of the food. It then calls the reset() method to spawn it at a random location.

2.1.2.2 Food::~Food ( )

Destructor that destroys a piece of food.

This method destroys a food object.

2.1.3 Member Function Documentation

2.1.3.1 void Food::draw (sf::RenderWindow & *window*)

Draws the piece of food.

Parameters

in	*window*	The sf::RenderedWindow of the game.

This method draws the body of the food to the games window.

2.1.3.2 sf::Vector2f Food::getPosition ( )

Returns the foods position.

Returns

A sf::Vector2f which represents the foods position.

This method returns the x and y coordinates of the bodys position in a Vector.

2.1.3.3 void Food::reset ( )

Resets the foods position to a random position.

This method calls the getRandom method to calculate random x and y coordinates and then sets the bodys position to them.

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

- src/Food.h
- src/Food.cpp

2.2 Game Class Reference

Contains the gameloop.

`#include <Game.h>`

Public Member Functions

- Game ()
  Constructor which creates a game.
- ∼Game ()
  Destructor which destroys a game.

Static Public Member Functions

- static void gameLoop (sf::Texture textureSegment, sf::Texture textureFood, sf::Font font, int &highScore, sf↩ ::Music ∗music, sf::SoundBuffer bufferCollision, sf::SoundBuffer bufferEat)
  The actual gameloop.

2.2.1 Detailed Description

Contains the gameloop.

This class is called by the main.cpp and is the actual game. It contains the gameloop which loops until the game ends.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Game::Game ( )

Constructor which creates a game.

This method creates a game.

2.2.2.2 Game::∼Game ( )

Destructor which destroys a game.

This method destroys a game.

2.2.3 Member Function Documentation

2.2.3.1 void Game::gameLoop (sf::Texture *textureSegment,* sf::Texture *textureFood,* sf::Font *font,* int & *highScore,* sf::Music ∗ *music,* sf::SoundBuffer *bufferCollision,* sf::SoundBuffer *bufferEat*) [static]

The actual gameloop.

Parameters

in	*textureSegment*	The texture of one segment of a snake.
in	*textureFood*	The texture of a food.
in	*font*	The font for the text.
in	*highScore*	The current highscore.
in	*music*	The games background music.
in	*bufferCollision*	The sound for a collision.
in	*bufferEat*	The sound for eating food.

At first this method sets up the required objects, variables and the window. It contains a loop that loops until the x-button is clicked. In the loop it calls methods, checks conditions and draws everything to the window.

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

- src/Game.h
- src/Game.cpp

2.3 Snake Class Reference

The snake which is controled by the player.

`#include <Snake.h>`

Public Member Functions

- Snake (const sf::Vector2u winSize, float sp, sf::Texture ∗texture)
  Constructor which creates a snake.
- void changeDirection ()
  Changes the snakes direction.
- void reset ()
  Resets the snake.
- bool checkCollison ()
  Checks if the snake collides with the wall or its tail.
- bool eats (sf::Vector2f foodPos)
  Checks if the snake eats a piece of food.
- void addSegments (int number)
  Checks if the snake eats a piece of food.
- void update ()
  Updates the position of the snake.
- void draw (sf::RenderWindow &window)
  Draws all the segments of the snake.
- sf::Vector2f getSegmentSize ()
  Returns the size of one segment.
- ∼Snake ()
  Destructor that destroys a snake.

2.3.1 Detailed Description

The snake which is controled by the player.

This class represents the snake that is controled by the player. It starts a small rectangle and grows its tail whenever it eats a piece of food. It's reset when it hits either the wall or its own tail. The snake is represented by a std::vector of segments.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Snake::Snake (const sf::Vector2u *winSize,* float *sp,* sf::Texture * *texture*)

Constructor which creates a snake.

Parameters

in	*winSize*	The size of the games window.
in	*texture*	The texture of one segment of the snake.

This method creates a snake. It sets the texture, size, origin and location of the snakes first segment and inserts it into the std::vector (tail).

2.3.2.2 Snake::~Snake ( )

Destructor that destroys a snake.

This method destroys a snake object.

2.3.3 Member Function Documentation

2.3.3.1 void Snake::addSegments (int *number*)

Checks if the snake eats a piece of food.

Parameters

in	*number*	The number of segments to add.

This method adds the amount of new segments to the std::vector (tail).

2.3.3.2 void Snake::changeDirection ( )

Changes the snakes direction.

This method changes the direction in which the snakes head (tail[0]) moves depending on the key that is pressed by the player.

2.3.3.3 bool Snake::checkCollison ( )

Checks if the snake collides with the wall or its tail.

Returns

A boolean that states if the snake collides.

This method calculates the distance of the snakes head (tail[0]) to every other segment in the std::vector (tail). If the segments collide, the method returns true. Then it checks if the snake hits the border of the window. If it does, the method returns true.

2.3.3.4 void Snake::draw (sf::RenderWindow & *window*)

Draws all the segments of the snake.

Parameters

in	*window*	The sf::RenderedWindow of the game.

This method iterates through the std::vector (tail) and draws all its segments to the window.

2.3.3.5 bool Snake::eats (sf::Vector2f *foodPos*)

Checks if the snake eats a piece of food.

Parameters

in	*foodPos*	The position of the food.

Returns

A boolean that states if the snake eats the food.

This method calculates the distance between the snakes head (tail[0]) and the given piece of food. If they collide, the method returns true;

2.3.3.6 sf::Vector2f Snake::getSegmentSize ( )

Returns the size of one segment.

Returns

A sf::Vector2f which represents the segment size.

This method returns the size of one segment of the std::vector (tail).

2.3.3.7 void Snake::reset ( )

Resets the snake.

This method first clears the std::vector (tail). It then inserts only the starting segment which has its position at the middle of the window.

2.3.3.8 void Snake::update ( )

Updates the position of the snake.

This method reversely iterates through all the segments of the std::vector (tail) except the head (tail[0]) and sets its position to the position of the next segment. Then it moves the head (tail[0]) by the direction.

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

- src/Snake.h
- src/Snake.cpp

Index