Snake

1.0

Generated by Doxygen 1.9.6

Chapter 1

Snake-CP-SemIII

Snake game made for semIII computer programming project

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

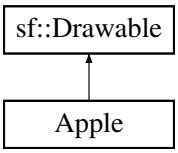
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Apple Class Reference

Inheritance diagram for Apple:



Public Member Functions

- **Apple** ()

  default constructor, that generates snakes food at random location
- void **changePosition** ()

  function, that changes the position of apple to new random position
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override

  draws an apple to the target

Friends

- void **checkCollision** (Snake &snake, Apple &apple, Board &board)

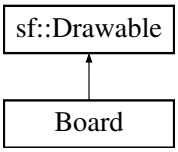
  function, that checks, if snakes head touched apple

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

- apple.h
- apple.cpp

5.2 Board Class Reference

Inheritance diagram for Board:



Public Member Functions

- **Board** ()
  default constructor, that generates gui
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override
  draws a gui to the target
- int **drawScore** ()
  function that updates a score to print, and then prints it to the gui
- void setScore (int value)
- int **getScore** ()
  function returning score

Friends

- void **checkCollision** (Snake &snake, Apple &apple, Board &board)
  function that checks if snakes head touches apple

5.2.1 Member Function Documentation

5.2.1.1 setScore()

```
void Board::setScore (
            int value )
```

function that sets a score variable. It takes as an argument a value to replace

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

- board.h
- board.cpp

5.3 Game Class Reference

class that runs the game

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

Public Member Functions

- Game ()
- void run ()

5.3.1 Detailed Description

class that runs the game

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Game()

```
Game::Game ( )
```

default constructor that creates a window of size specified in options file, and creates all objects that belong to it

5.3.3 Member Function Documentation

5.3.3.1 run()

```
void Game::run ( )
```

core function that starts a game

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

- game.h
- game.cpp

5.4 Options Class Reference

Public Member Functions

- **Options** (Options const &)=delete

  singletons are not copyable
- void **loadFromFile** (std::string filename)

  loading a variables from specific file
- void **operator=** (Options const &)=delete

  singletons are not assignable

Static Public Member Functions

- static Options & **getInstance** ()

  the only way to create a instance of the class

Public Attributes

- int **windowSizeX**
- int **windowSizeY**
- int **snakeSpeed**

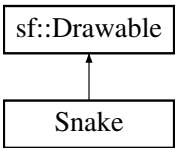
The documentation for this class was generated from the following files:

- options.h
- options.cpp

5.5 Snake Class Reference

`#include <snake.h>`

Inheritance diagram for Snake:



Public Member Functions

- **Snake** (Snake &&snake)

  default snake constructor, that generates 3 beggining connected rectangles
- bool update ()
- sf::Vector2f & **getHeadPosition** ()

  function that returns a reference to the head position of a snake
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override

  draws a snake to the target

Friends

- void checkCollision (Snake &snake, Apple &apple, Board &board)
- std::ostream & operator<< (std::ostream &os, const Snake &snake)

5.5.1 Detailed Description

This is snake Class It is inherited from sf::Drawable class, so it can override draw function

5.5.2 Member Function Documentation

5.5.2.1 update()

```
bool Snake::update ( )
```

function that updates a position of all snake tiles. It returns a boolean that shows if snake hit itsself or not.

5.5.3 Friends And Related Function Documentation

5.5.3.1 checkCollision

```
void checkCollision (
            Snake & snake,
            Apple & apple,
            Board & board )  [friend]
```

checks if snakes head is in the same place as apple. If yes, then change apple position, and increment score in board

5.5.3.2 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Snake & snake )  [friend]
```

This function prints out a snake elements positions for debugging purposes

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

- snake.h
- snake.cpp

Chapter 6

File Documentation

6.1 apple.h

```
00001 #ifndef APPLE_H_
00002 #define APPLE_H_
00003 #include <SFML/System/Vector2.hpp>
00004 #include <SFML/Graphics/Drawable.hpp>
00005 class Snake;
00006 class Board;
00007
00008 class Apple : public sf::Drawable
00009 {
00010     sf::Vector2f position;
00011
00012 public:
00014     Apple();
00016     void changePosition();
00018     void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00020     friend void checkCollision(Snake &snake, Apple &apple, Board &board);
00021 };
00022
00023 #endif
```

6.2 board.h

```
00001 #ifndef BOARD_H_
00002 #define BOARD_H_
00003 #include "apple.h"
00004 #include "options.h"
00005 #include "snake.h"
00006 #include <SFML/Graphics/Drawable.hpp>
00007 #include <SFML/Graphics/Font.hpp>
00008 #include <SFML/Graphics/RectangleShape.hpp>
00009 #include <SFML/Graphics/Text.hpp>
00010 #include <string>
00011
00012 class Board : public sf::Drawable {
00016   sf::RectangleShape area;
00017
00018   sf::Font font;
00019   sf::Text text;
00020   sf::Text scoreText;
00021   int score;
00022
00023 public:
00025   Board();
00027   void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00029   int drawScore();
00032   void setScore(int value);
00034   int getScore();
00036   friend void checkCollision(Snake &snake, Apple &apple, Board &board);
00037 };
00038
00039 #endif
```

6.3 functions.h

```
00001 #ifndef FUNC_H_
00002 #define FUNC_H_
00003 #include <random>
00004 #include "apple.h"
00005 #include "snake.h"
00006
00008 int randomInt(int min, int max);
00009
00010 #endif
```

6.4 game.h

```
00001 #ifndef GAME_H_
00002 #define GAME_H_
00003 #include <SFML/Graphics/RenderWindow.hpp>
00004 #include <SFML/Window/Event.hpp>
00005 #include "snake.h"
00006 #include "board.h"
00007 #include "apple.h"
00008 #include "options.h"
00009 // -*- lsst-c++ -*-
00010
00014 class Game
00015 {
00019     sf::RenderWindow window;
00024     Snake snake;
00028     Board board;
00032     Apple apple;
00033 public:
00037     Game();
00041     void run();
00042 };
00043
00044 #endif
```

6.5 options.h

```
00001 #ifndef OPT_H_
00002 #define OPT_H_
00003
00004 #include <string>
00005
00006 class Options
00007 {
00008 public:
00010     static Options &getInstance()
00011     {
00012         static Options instance;
00013
00014         return instance;
00015     }
00016
00017 private:
00018     Options() {}
00019
00020 public:
00022     Options(Options const &) = delete;
00024     void loadFromFile(std::string filename);
00026     void operator=(Options const &) = delete;
00027     int windowSizeX;
00028     int windowSizeY;
00029     int snakeSpeed;
00030 };
00031 #endif
```

6.6 snake.h

```
00001 #ifndef SNAKE_H_
00002 #define SNAKE_H_
00003 #include <SFML/Graphics/Drawable.hpp>
00004 #include <SFML/Graphics/RectangleShape.hpp>
00005 #include <SFML/Graphics/RenderTarget.hpp>
00006 #include <SFML/System/Vector2.hpp>
```

```
00007 #include <SFML/Window/Keyboard.hpp>
00008 #include <iostream>
00009 #include <vector>
00010
00011 class Apple;
00012
00013 class Board;
00018 class Snake : public sf::Drawable {
00022   enum class Direction { UP, RIGHT, DOWN, LEFT };
00023
00027   Direction currentDirection;
00032   std::vector<sf::Vector2f> segments;
00036   sf::RectangleShape rect;
00041   sf::Vector2f lastSegmentPosition;
00042
00043 public:
00045   Snake(Snake &&snake) { currentDirection = std::move(snake.currentDirection); }
00046   Snake();
00049   bool update();
00051   sf::Vector2f &getHeadPosition();
00053   void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00056   friend void checkCollision(Snake &snake, Apple &apple, Board &board);
00060   friend std::ostream &operator«(std::ostream &os, const Snake &snake);
00061 };
00062
00063 #endif
```

Index