Shape Shooters

Generated by Doxygen 1.9.5

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 Bullet Class Reference . . . 5

3.1.1 Constructor & Destructor Documentation . . . 5

3.1.1.1 Bullet() [1/2] . . . 5

3.1.1.2 Bullet() [2/2] . . . 5

3.1.1.3 ~Bullet() . . . 6

3.1.2 Member Function Documentation . . . 6

3.1.2.1 getBounds() . . . 6

3.1.2.2 update() . . . 6

3.2 Enemy Class Reference . . . 6

3.2.1 Constructor & Destructor Documentation . . . 6

3.2.1.1 Enemy() . . . 7

3.2.1.2 ~Enemy() . . . 7

3.2.2 Member Function Documentation . . . 7

3.2.2.1 getBounds() . . . 7

3.2.2.2 update() . . . 7

3.3 Game Class Reference . . . 7

3.3.1 Constructor & Destructor Documentation . . . 8

3.3.1.1 Game() . . . 8

3.3.1.2 ~Game() . . . 8

3.3.2 Member Function Documentation . . . 8

3.3.2.1 render() . . . 8

3.3.2.2 renderGUI() . . . 8

3.3.2.3 run() . . . 9

3.3.2.4 updateBullets() . . . 9

3.3.2.5 updateCollision() . . . 9

3.3.2.6 updateEnemies() . . . 9

3.3.2.7 updateGUI() . . . 9

3.3.2.8 updateInput() . . . 10

3.3.2.9 updatePollEvents() . . . 10

3.3.2.10 updateWar() . . . 10

3.4 Player Class Reference . . . 11

3.4.1 Constructor & Destructor Documentation . . . 11

3.4.1.1 Player() . . . 11

3.4.1.2 ~Player() . . . 11

3.4.2 Member Function Documentation . . . 11

3.4.2.1 getPos() . . . 11

3.4.2.2 move() 12
3.4.2.3 setPosition() 12

4 File Documentation **13**

4.1 Bullet.h 13
4.2 Enemy.h 13
4.3 Game.h 14
4.4 Player.h 15

Index **17**

Chapter 1

Class Index

1.1 Class List

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

Bullet 5
Enemy 6
Game 7
Player 11

Chapter 2

File Index

2.1 File List

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

Bullet.h . . . . . ??
Enemy.h . . . . . ??
Game.h . . . . . ??
Player.h . . . . . ??

Chapter 3

Class Documentation

3.1 Bullet Class Reference

Public Member Functions

- Bullet ()
- Bullet (sf::Texture *texture, float positionX, float positionY, float coordinateX, float coordinateY, float movementVelocity)
- virtual ~Bullet ()
- sf::FloatRect getBounds () const
- void update ()
- void **render** (sf::RenderTarget *target)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Bullet() [1/2]

```
Bullet::Bullet ( )  [default]
```

Bullet Default Constructor

Bullet constructor

3.1.1.2 Bullet() [2/2]

```
Bullet::Bullet (
            sf::Texture * texture,
            float positionX,
            float positionY,
            float coordinateX,
            float coordinateY,
            float movementVelocity )
```

Bullet constructor

3.1.1.3 ~Bullet()

```
Bullet::~Bullet ( )  [virtual], [default]
```

Bullet destructor

3.1.2 Member Function Documentation

3.1.2.1 getBounds()

```
sf::FloatRect Bullet::getBounds ( ) const
```

Bullet accessors getBounds() determines the boundaries of the bullet

3.1.2.2 update()

```
void Bullet::update ( )
```

Bullet public functions including update and render

Bullet public functions implementation How we move the bullet

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

- Bullet.h
- Bullet.cpp

3.2 Enemy Class Reference

Public Member Functions

- Enemy (float positionX, float positionY)
- virtual ~Enemy ()
- sf::FloatRect getBounds () const
- const int & **getPoints** () const
- const int & **getDamage** () const
- void update ()
- void **render** (sf::RenderTarget *target)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Enemy()

```
Enemy::Enemy (
            float positionX,
            float positionY )
```

Enemy constructor

3.2.1.2 ~Enemy()

```
Enemy::~Enemy ( )  [virtual], [default]
```

Enemy destructor

3.2.2 Member Function Documentation

3.2.2.1 getBounds()

```
sf::FloatRect Enemy::getBounds ( ) const
```

Enemy accessor

3.2.2.2 update()

```
void Enemy::update ( )
```

Enemy public functions Like Player, enemy is needed to get updated and rendered which is defined here via update() and render() respectively.

Enemy public functions

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

- Enemy.h
- Enemy.cpp

3.3 Game Class Reference

Public Member Functions

- Game ()
- virtual ~Game ()
- void run ()
- void updatePollEvents ()
- void updateInput ()
- void updateGUI ()
- void updateCollision ()
- void updateBullets ()
- void updateEnemies ()
- void updateWar ()
- void **update** ()
- void renderGUI ()
- void **renderUniverse** ()
- void render ()

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Game()

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

Game constructor

Game Constructor

3.3.1.2 ~Game()

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

Game destructor

Game Destructor To avoid memory leaks after the game ends Delete textures

Delete bullets

Delete enemies

3.3.2 Member Function Documentation

3.3.2.1 render()

```
void Game::render ( )
```

Draw universe

Drawing everything here

Render the player inside the window

Rendering game over screen here

3.3.2.2 renderGUI()

```
void Game::renderGUI ( )
```

The order of rendering is extremely important!

3.3.2.3 run()

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

Game public functions GUIs, universe, collisions, bullets, enemies and etc get updated here

Game functions implementation

3.3.2.4 updateBullets()

```
void Game::updateBullets ( )
```

Managing bullets when going toward top of the screen

Delete bullet from memory and erase them as well.

3.3.2.5 updateCollision()

```
void Game::updateCollision ( )
```

When player collides with the corners of the screen.

Left corner of the screen

Right corner of the screen

Top corner of the screen

Bottom corner of the screen

3.3.2.6 updateEnemies()

```
void Game::updateEnemies ( )
```

Enemies spawn here

Updating enemies here

Managing enemies when going toward bottom of the screen

Delete enemy from memory and erase them as well.

If enemy collides with player then it (enemy) still will be erased

3.3.2.7 updateGUI()

```
void Game::updateGUI ( )
```

Here we update player's GUI At healthPointPercent, we get the percentage of health point

3.3.2.8 updateInput()

```
void Game::updateInput ( )
```

Move Player

MOVE TO THE LEFT

MOVE TO THE RIGHT

MOVE FORWARD

MOVE BACKWARD

When left button of mouse is pressed and player can attack

3.3.2.9 updatePollEvents()

```
void Game::updatePollEvents ( )
```

Two different ways to close the window:

1. by pressing the close button
2. by pressing the Esc button on the keyboard

3.3.2.10 updateWar()

```
void Game::updateWar ( )
```

Check the collision between enemy and bullet

Getting points when bullet collides with enemy

Delete and erase enemy after collision

Delete and erase bullet after collision

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

- Game.h
- Game.cpp

3.4 Player Class Reference

Public Member Functions

- Player ()
- virtual ~Player ()
- const sf::Vector2f & getPos () const
- sf::FloatRect **getBounds** () const
- const int & **getHealthPoint** () const
- const int & **getHealthPointMax** () const
- void setPosition (float x, float y)
- void **reduceHealthPoint** (const int value)
- void move (float coordinateX, float coordinateY)
- bool **canAttack** ()
- void **updateAttack** ()
- void **update** ()
- void **render** (sf::RenderTarget &target)

3.4.1 Constructor & Destructor Documentation

3.4.1.1 Player()

```
Player::Player ( )
```

Player constructor

3.4.1.2 ~Player()

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

Player destructor

3.4.2 Member Function Documentation

3.4.2.1 getPos()

```
const sf::Vector2f & Player::getPos ( ) const
```

Player accessors including getPos() for position, getBounds() for getting boundaries of the player. getHealthPoint() and getHealthPointMax() for getting the health point

Player accessor implementation

3.4.2.2 move()

```
void Player::move (
            float coordinateX,
            float coordinateY )
```

Player public functions including move() to move the player, canAttack() and updateAttack() to handle when the attacking situations Other functions are update() to update the player as well as render() to render.

Player public functions

3.4.2.3 setPosition()

```
void Player::setPosition (
            float x,
            float y )
```

Player modifiers Set the position of the player Set and reduce the health point of the player

Player modifier implementation

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

- Player.h
- Player.cpp

Chapter 4

File Documentation

4.1 Bullet.h

```
1 #ifndef BULLET_H
2 #define BULLET_H
3
4 #include<SFML/Graphics.hpp>
5 #include<iostream>
6
7 class Bullet {
8 private:
12      sf::Sprite shape;
13
14      sf::Vector2f direction;
15      float movementVelocity{};
16
17 public:
21      Bullet();
25      Bullet(sf::Texture* texture, float positionX, float positionY, float coordinateX, float coordinateY,
        float movementVelocity);
29      virtual ~Bullet();
30
35      sf::FloatRect getBounds() const;
36
40      void update();
41      void render(sf::RenderTarget* target);
42
43 };
44
45
46
47 #endif
```

4.2 Enemy.h

```
1 #ifndef ENEMY_H
2 #define ENEMY_H
3
4 #include<SFML/Graphics.hpp>
5 #include<iostream>
6
7
8 class Enemy {
9 private:
10      unsigned pointCount;
11      sf::CircleShape shape;
12      float speed;
13      int healthPoint;
14      int healthPointMax;
15      int damage;
16      int points;
17
22      void initializeVariables();
23      void initializeShape();
24 public:
28      Enemy(float positionX, float positionY);
29
```

```
33      virtual ~Enemy();
34
38      sf::FloatRect getBounds() const;
39      const int& getPoints() const;
40      const int& getDamage() const;
41
46      void update();
47      void render(sf::RenderTarget* target);
48  };
49
50
51  #endif
```

4.3 Game.h

```
1   #ifndef GAME_H
2   #define GAME_H
3
4   #include "Player.h"
5   #include "Bullet.h"
6   #include "Enemy.h"
7   #include<map>
8   #include<string>
9   #include<sstream>
10
11
12  class Game {
13  private:
17      sf::RenderWindow* window{};
18
24      std::map<std::string, sf::Texture*> textures;
25      std::vector<Bullet*> bullets;
26
30      sf::Font font;
31      sf::Text pointText;
32      sf::Text gameOverText;
33
37      sf::Texture universeBackgroundTexture;
38      sf::Sprite universeBackground;
39
43       unsigned points;
44
48      Player* player{};
49
53       sf::RectangleShape playerHealthPointBar;
54       sf::RectangleShape playerHealthPointBarBackground;
55
59      float spawnTimer{};
60      float spawnTimerMax{};
61      std::vector<Enemy*> enemies;
62
63
68      void initializeWindow();
69      void initializeTextures();
70      void initializeGUI();
71      void initializeUniverse();
72      void initializeSystems();
73      void initializePlayer();
74      void initializeEnemy();
75
76  public:
80      Game();
84      virtual ~Game();
85
86
87
92      void run();
93      void updatePollEvents();
94      void updateInput();
95      void updateGUI();
96      void updateCollision();
97      void updateBullets();
98      void updateEnemies();
99      void updateWar();
100      void update();
101      void renderGUI();
102      void renderUniverse();
103      void render();
104
105 };
106
107
108 #endif
```

4.4 Player.h

```
1 #ifndef PLAYER_H
2 #define PLAYER_H
3
4 #include<SFML/Graphics.hpp>
5 #include<iostream>
6
7 class Player {
8 private:
12     sf::Sprite sprite;
13     sf::Texture texture;
14
15     float movementVelocity{};
16
17     float attackControl{};
18     float attackControlMax{};
19
20     int healthPoint;
21     int healthPointMax;
22
27     void initializeVariables();
28     void initializeTexture();
29     void initializeSprite();
30
31 public:
35     Player();
36
40     virtual ~Player();
41
46     const sf::Vector2f& getPos() const;
47     sf::FloatRect getBounds() const;
48     const int& getHealthPoint() const;
49     const int& getHealthPointMax() const;
50
56     //void setPosition(sf::Vector2f position);
57     void setPosition(float x, float y);
58     //void setHealthPoint(const int hp);
59     void reduceHealthPoint(const int value);
60
65     void move(float coordinateX, float coordinateY);
66     bool canAttack();
67     void updateAttack();
68     void update();
69     void render(sf::RenderTarget& target);
70
71 };
72
73
74 #endif
```

Index

~Bullet
    Bullet, 5
~Enemy
    Enemy, 7
~Game
    Game, 8
~Player
    Player, 11

Bullet, 5
    ~Bullet, 5
    Bullet, 5
    getBounds, 6
    update, 6

Enemy, 6
    ~Enemy, 7
    Enemy, 6
    getBounds, 7
    update, 7

Game, 7
    ~Game, 8
    Game, 8
    render, 8
    renderGUI, 8
    run, 8
    updateBullets, 9
    updateCollision, 9
    updateEnemies, 9
    updateGUI, 9
    updateInput, 9
    updatePollEvents, 10
    updateWar, 10
getBounds
    Bullet, 6
    Enemy, 7
getPos
    Player, 11

move
    Player, 11

Player, 11
    ~Player, 11
    getPos, 11
    move, 11
    Player, 11
    setPosition, 12

render
    Game, 8
renderGUI
    Game, 8
run
    Game, 8

setPosition
    Player, 12

update
    Bullet, 6
    Enemy, 7
updateBullets
    Game, 9
updateCollision
    Game, 9
updateEnemies
    Game, 9
updateGUI
    Game, 9
updateInput
    Game, 9
updatePollEvents
    Game, 10
updateWar
    Game, 10