Simple Game

AUTHOR: Mateusz Adamek
Mail:madamek@studnet.agh.edu.pl
Index:305117
Version 1.0.1
Mon Jan 18 2021

Project assumptions

The main purpose of the project was to make a simple game with GUI-Graphical user interface. The Graphic interface was made using SFML. The main goal of the game is to win a prize that is on the board. Obstacles make it difficult to achieve your goal

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

Class Documentation

Action Class Reference

Public Member Functions

- void **MainMenu_draw** (sf::RenderWindow &window, int &check, sf::Event &event, **Menu** &menu)
- void **Game_draw** (sf::RenderWindow &window, **Ball** &ball, **Ball** &ball2, **Ball** &ball3, **Ball** &ball4, **Ball** &ball5, **Ball** &ball6, **Pawn** &player, **Meta** &meta, **End_menu** &the_end2, int &check)
- void **Options_menu** (sf::RenderWindow &window, sf::Event &event, int &check, **Options** &options)
- void **Display_level** (sf::RenderWindow &window, **Ball** &ball, **Ball** &ball2, **Ball** &ball3, **Ball** &ball4, **Ball** &ball5, **Ball** &ball6, sf::Event event, **Level_menu** &finish, std::vector< std::vector< int >> &level_hard, std::vector< std::vector< int >> &level_medium, std::vector< std::vector< int >> &level_easy, int &check)
- void **Music_set** (sf::RenderWindow &window, sf::Event &event, int &check, std::string &a, **Music_choise_menu** &music_choise, std::vector< std::string > &utwory, sf::Music &music)
- void **Color_set_menu** (sf::RenderWindow &window, sf::Event &event, **Pawn** &player, int &check, **Color_Menu** &color)
- void **Exit_menu** (sf::RenderWindow &window, **Pawn** &player, int &check, sf::Event &event, **End_menu** &the_end2)

Member Function Documentation

void Action::Color_set_menu (sf::RenderWindow & *window*, sf::Event & *event*, Pawn & *player*, int & *check*, Color_Menu & *color*)

Color_set_menu this is a method which allow choise color of our player/pawn

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	sf::Event ,this is part of SFML library, Event allow us to interaction beetween game and user using keyboards.
3	transfers the **Pawn** object that will be displayed in our game window
4	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called
5	transfers the Color_menu object that will be displayed in our window

void Action::Display_level (sf::RenderWindow & *window*, Ball & *ball*, Ball & *ball2*, Ball & *ball3*, Ball & *ball4*, Ball & *ball5*, Ball & *ball6*, sf::Event *event*, Level_menu & *finish*, std::vector< std::vector< int >> & *level_hard*, std::vector< std::vector< int >> & *level_medium*, std::vector< std::vector< int >> & *level_easy*, int & *check*)

Display_level this is a method that allows you levels of our game to choose

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	transfers the ball object that will be displayed in our game window
3	transfers the ball2 object that will be displayed in our game window
4	transfers the ball3 object that will be displayed in our game window
5	transfers the ball4 object that will be displayed in our game window
6	transfers the ball5 object that will be displayed in our game window
7	transfers the ball6 object that will be displayed in our game window
8	sf::Event ,this is part of SFML library, Event allow us to interaction beetween

	game and user using keyboards.
9	transfers the **Level_menu** object that will be displayed in our window
10	We pass a vector that contains, properties for LVL HARD
11	We pass a vector that contains, properties for LVL MEDIUM
12	We pass a vector that contains, properties for LVL EASY
13	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called

void Action::Exit_menu (sf::RenderWindow & *window*, Pawn & *player*, int & *check*, sf::Event & *event*, End_menu & *the_end2*)

Exit_menu this is the menu which will be displayed when the player finished the game or will be killed from the object

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	transfers the **Pawn** object that will be displayed in our game window
3	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called
4	sf::Event ,this is part of SFML library, Event allow us to interaction beetween game and user using keyboards.
5	transfers the **End_menu** object that will be displayed in our window

void Action::Game_draw (sf::RenderWindow & *window*, Ball & *ball*, Ball & *ball2*, Ball & *ball3*, Ball & *ball4*, Ball & *ball5*, Ball & *ball6*, Pawn & *player*, Meta & *meta*, End_menu & *the_end2*, int & *check*)

Game_draw it is the method which is responsible for start and display game on our window,

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	transfers the ball object that will be displayed in our game window
3	transfers the ball2 object that will be displayed in our game window
4	transfers the ball3 object that will be displayed in our game window
5	transfers the ball4 object that will be displayed in our game window
6	transfers the ball5 object that will be displayed in our game window
7	transfers the ball6 object that will be displayed in our game window
8	transfers the **Pawn** object that will be displayed in our game window
9	transfers the **Meta** object that will be displayed in our game window
10	transfers the **End_menu** object that will be displayed in our game window
11	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called

void Action::MainMenu_draw (sf::RenderWindow & *window*, int & *check*, sf::Event & *event*, Menu & *menu*)

MainMenu_draw display Main **Menu** in windows, we provide the parameters through references, the function accepts the following parameters

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	int check, this is the variable that is responsible for exporting the method
3	sf::Event ,this is part of SFML library, Event allow us to interaction beetween game and user using keyboards.
4	**Menu**, object of class **Menu** which will be displaying

void Action::Music_set (sf::RenderWindow & *window*, sf::Event & *event*, int & *check*, std::string & *a*, Music_choise_menu & *music_choise*, std::vector< std::string > & *utwory*, sf::Music & *music*)

Music_set A method that allows you to select background music

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	sf::Event ,this is part of SFML library, Event allow us to interaction beetween game and user using keyboards.
3	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called
4	in this variable we set currently playing background music
5	transfers the Music_choice_menu object that will be displayed in our window
6	We pass a vector that contains, properties for avaliable song to choice.
7	sf::Music , we pass object of SFML Music,

void Action::Options_menu (sf::RenderWindow & *window*, sf::Event & *event*, int & *check*, Options & *options*)

Options_menu this is a method that allows you to view variable from the options menu

Parameters

1	sf::RenderWindow , we must get window fromm SFML library
2	sf::Event ,this is part of SFML library, Event allow us to interaction beetween game and user using keyboards.
3	int check, this is the variable that is responsible for changing the method, if check will be changed, other funvtion from **Action** class will be called
4	transfers the **Options** object that will be displayed in our window

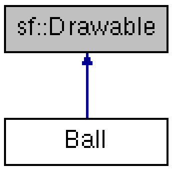
---

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

- Action.h
- Action.cpp

Ball Class Reference

Inheritance diagram for Ball:



Public Member Functions

- **Ball** (float t_X, float t_Y, float velocity_X, float velocity_Y)
- void **update** ()
- float **left** ()
- float **right** ()
- float **top** ()
- float **bottom** ()
- void **change_level** (float x, float y)

Constructor & Destructor Documentation

Ball::Ball (float *t_X*, float *t_Y*, float *velocity_X*, float *velocity_Y*)

this is the contructor of our **Ball**

Parameters

1	the X axe of size of our ball (float)
2	the Y axe of size of our ball (float)
3	the speed of X direction (float)
4	the speed of Y direction (float)

Member Function Documentation

float Ball::bottom ()

void **bottom()** this function calculated left edge of our ball

Returns

coordinates of bottom edge of ball

void Ball::change_level (float *x*, float *y*)

This function allow change the level of our game,

Parameters

1	the speed of X direction (float)
2	the speed of Y direction (float)

float Ball::left ()

void **left()** this function calculated left edge of our ball

Returns

coordinates of left edge of ball

float Ball::right ()

void **right()** this function calculated right edge of our ball

Returns

coordinates of right edge of ball

float Ball::top ()

void **top()** this function calculated top edge of our ball

Returns

coordinates of top edge of ball

void Ball::update ()

update is the function which refresh object on the window, no parameters

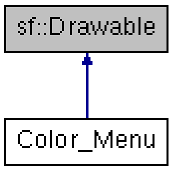
---

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

- Ball.h
- Ball.cpp

Color_Menu Class Reference

Inheritance diagram for Color_Menu:



Public Member Functions

- **Color_Menu** (float width, float height)
  constructor whith parameterf of Color ***Menu*** *@params1 width of our display window (float) @params2 height f our display window (float)*

- void **MoveLeft** ()
- void **MoveRight** ()
- int **GetPressedItem** ()

Member Function Documentation

int Color_Menu::GetPressedItem () `[inline]`

Function which return which label was be choose.

Returns

number of level which will be set highlighting

void Color_Menu::MoveLeft ()

Function which allow move left of highlighting. element of the left will be light on other color than rest element.

void Color_Menu::MoveRight ()

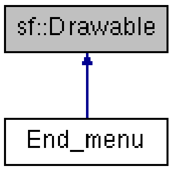
Function which allow move right of highlighting. element of the left will be light on other color than rest element.

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

- Color_Menu.h
- Color_Menu.cpp

End_menu Class Reference

Inheritance diagram for End_menu:



Public Member Functions

- **End_menu** (float width, float height)
- void **MoveLeft** ()
- void **MoveRight** ()
- int **GetPressedItem** ()
- void **set_finish** (bool which_finish)

Constructor & Destructor Documentation

End_menu::End_menu (float *width*, float *height*)

The constructor of class @params1 width of window to display(float) @params2 height of window to display(float)

Member Function Documentation

int End_menu::GetPressedItem () `[inline]`

Function which return which label was be choice.

Returns

number of level which will be set highlighting

void End_menu::MoveLeft ()

Function which allow move left of highlighting. element of the left will be light on other color than rest element.

void End_menu::MoveRight ()

Function which allow move right of highlighting. element of the left will be light on other color than rest element.

void End_menu::set_finish (bool *which_finish*)

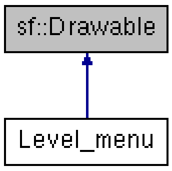
This function set which end menu will be displayed @params1 transform (bool) variable , TRUE means that will be displayed menu that YOU WIN the game, if set FALSE will be displayed LOOSE GAME MENU

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

- End_menu.h
- End_menu.cpp

Level_menu Class Reference

Inheritance diagram for Level_menu:



Public Member Functions

- **Level_menu** (float width, float height)
- void **MoveLeft** ()
- void **MoveRight** ()
- int **GetPressedItem** ()

Constructor & Destructor Documentation

Level_menu::Level_menu (float *width*, float *height*)

constructor whith parameterf of **Level_menu** @params1 width of our display window (float) @params2 height f our display window (float)

Member Function Documentation

int Level_menu::GetPressedItem () `[inline]`

Function which return which label was be choose.

Returns

number of level which will be set highlighting

void Level_menu::MoveLeft ()

Function which allow move left of highlighting. element of the left will be light on other color than rest element.

void Level_menu::MoveRight ()

Function which allow move right of highlighting. element of the left will be light on other color than rest element.

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

- Level_menu.h
- Level_menu.cpp

Menu Class Reference

Public Member Functions

- **Menu** (float width, float height)
- void **draw** (sf::RenderWindow &window)
- void **MoveUp** ()
- void **MoveDown** ()
- void **close** (sf::RenderWindow &window)
- int **GetPressedItem** ()

Constructor & Destructor Documentation

Menu::Menu (float *width*, float *height*)

The constructor of class @params1 width of window to display(float) @params2 height of window to display(float)

Member Function Documentation

void Menu::close (sf::RenderWindow & *window*)

Function which is responsibility for delete **Menu** object from main windows, all will be cleaned. @params1 sf::RenderWindow , part of SFML, main windows of the application

void Menu::draw (sf::RenderWindow & *window*)

Function which allow draw class **Menu** in SFML Redener window @parmas1 sf::RenderWindow , part of SFML, main windows of the application

int Menu::GetPressedItem () `[inline]`

Function which return which label was be choice.

Returns

number of level which will be set highlighting

void Menu::MoveDown ()

Function which allow move down of highlighting. element of the down will be light on other color than rest element.

void Menu::MoveUp ()

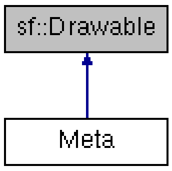
Function which allow move up of highlighting. element of the up will be light on other color than rest element.

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

- Menu.h
- Menu.cpp

Meta Class Reference

Inheritance diagram for Meta:



Public Member Functions

- **Meta** (float t_X, float t_Y)
- float **left** ()
- float **right** ()
- float **top** ()
- float **bottom** ()
- bool **isDestroyed** ()
- void **destroy** ()

Constructor & Destructor Documentation

Meta::Meta (float *t_X*, float *t_Y*)

Constructor of **Meta** @params1 width of window(float) @params2 hight of window(float)

Member Function Documentation

float Meta::bottom ()

void **bottom()** this function calculated left edge of our ball

Returns

coordinates of bottom edge of ball

void Meta::destroy ()

Set destroyed variable on TRUE,

bool Meta::isDestroyed ()

This function checks if the element has been damaged

Returns

True or False

float Meta::left ()

void **left()** this function calculated left edge of our ball

Returns

coordinates of left edge of ball

float Meta::right ()

void **right()** this function calculated right edge of our ball

Returns

coordinates of right edge of ball

float Meta::top ()

void **top()** this function calculated top edge of our ball

Returns

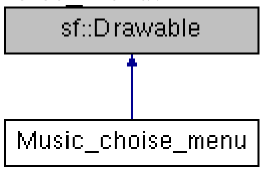
coordinates of top edge of ball

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

- meta.h
- meta.cpp

Music_choise_menu Class Reference

Inheritance diagram for Music_choise_menu:



Public Member Functions

- **Music_choise_menu** (float width, float height)
- void **MoveUp** ()
- void **MoveDown** ()
- int **GetPressedItem** ()

Constructor & Destructor Documentation

Music_choise_menu::Music_choise_menu (float *width*, float *height*)

The constructor of class @params1 width of window to display(float) @params2 height of window to display(float)

Member Function Documentation

int Music_choise_menu::GetPressedItem () `[inline]`

Function which return which label was be choice.

Returns

number of level which will be set highlighting

void Music_choise_menu::MoveDown ()

Function which allow move down of highlighting. element of the down will be light on other color than rest element.

void Music_choise_menu::MoveUp ()

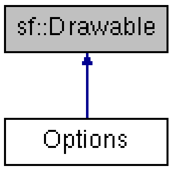
Function which allow move up of highlighting. element of the up will be light on other color than rest element.

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

- Music_choise_menu.h
- Music_choise_menu.cpp

Options Class Reference

Inheritance diagram for Options:



Public Member Functions

- **Options** (float width, float height)
- void **MoveUp** ()
- void **MoveDown** ()
- int **GetPressedItem** ()

Constructor & Destructor Documentation

Options::Options (float *width*, float *height*)

The constructor of class @params1 width of window to display(float) @params2 height of window to display(float)

Member Function Documentation

int Options::GetPressedItem () `[inline]`

Function which return which label was be choice.

Returns

number of level which will be set highlighting

void Options::MoveDown ()

Function which allow move down of highlighting. element of the down will be light on other color than rest element.

void Options::MoveUp ()

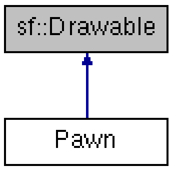
Function which allow move up of highlighting. element of the up will be light on other color than rest element.

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

- Options.h
- Options.cpp

Pawn Class Reference

Inheritance diagram for Pawn:



Public Member Functions

- **Pawn** (float t_X, float t_Y, float t_width, float t_high, float speed, sf::Color a)
- void **update** ()
- sf::Vector2f **getPosition** ()
- float **left** ()
- float **right** ()
- float **top** ()
- float **bottom** ()
- bool **isDestroyed** ()
- void **destroy** ()
- sf::Vector2f **getSize** ()
- void **change_color** (sf::Color a)
- void **position** ()

Constructor & Destructor Documentation

Pawn::Pawn (float *t_X*, float *t_Y*, float *t_width*, float *t_high*, float *speed*, sf::Color *a*)

Construcor of **Pawn** accepts the following parameters

Parameters

1	it is X coordinates when the **Pawn** started(float)
2	it is Y coordinates when the **Pawn** started(float)
3	the X axe of size of our ball (float)
4	the Y axe of size of our ball (float)
5	the speed of X direction (float)
6	the speed of Y direction (float)
7	sf::Color, set color of our object,

Member Function Documentation

float Pawn::bottom ()

void **bottom()** this function calculated left edge of our ball

Returns

coordinates of bottom edge of ball

void Pawn::change_color (sf::Color *a*)

This function set color of our **Pawn**

Parameters

1	sf::Color, get SFML color , to set color of our PAWN

void Pawn::destroy ()

Set destroyed variable on TRUE,

sf::Vector2f Pawn::getPosition ()

In this vector will we Position fo our **Pawn**

sf::Vector2f Pawn::getSize ()

In this vector will be size of our **Pawn**

bool Pawn::isDestroyed ()

This function checks if the element has been damaged, check that the player exist

Returns

True or False

float Pawn::left ()

void **left()** this function calculated left edge of our ball

Returns

coordinates of left edge of ball

void Pawn::position ()

This function set position of our **Pawn**

float Pawn::right ()

void **right()** this function calculated right edge of our ball

Returns

coordinates of right edge of ball

float Pawn::top ()

void **top()** this function calculated top edge of our ball

Returns

coordinates of top edge of ball

void Pawn::update ()

update function which refresh object on the window, no parameters

---

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

- Pawn.h
- Pawn.cpp

Project Files

https://github.com/AdamekMateusz/JPO2

Bibliography

https://www.youtube.com/watch?v=4Vg9d1pjL20&t=3s
https://www.youtube.com/watch?v=JIad3X3PX6o&list=PLk6mhiZKpyW4KRTZc8sc0aYOLFmTSLA7r
https://www.sfml-dev.org/documentation/2.5.1/