My Tetris

1.0

Generated by Doxygen 1.8.17

1 Class Index 1

1.1 Class List . . . 1

2 Class Documentation 3

2.1 tetris::Brick Class Reference . . . 3

2.1.1 Detailed Description . . . 4

2.1.2 Constructor & Destructor Documentation . . . 5

2.1.2.1 Brick() . . . 5

2.1.3 Member Function Documentation . . . 5

2.1.3.1 getPosition() . . . 5

2.1.3.2 inbound() . . . 5

2.1.3.3 move() . . . 6

2.1.3.4 render() . . . 6

2.1.3.5 rotate() . . . 6

2.2 tetris::Event Class Reference . . . 7

2.2.1 Detailed Description . . . 8

2.2.2 Member Enumeration Documentation . . . 8

2.2.2.1 EventType . . . 8

2.2.3 Constructor & Destructor Documentation . . . 9

2.2.3.1 Event() [1/3] . . . 9

2.2.3.2 Event() [2/3] . . . 9

2.2.3.3 Event() [3/3] . . . 10

2.3 tetris::Game Class Reference . . . 10

2.3.1 Detailed Description . . . 11

2.3.2 Constructor & Destructor Documentation . . . 11

2.3.2.1 Game() . . . 11

2.3.3 Member Function Documentation . . . 11

2.3.3.1 handlePlayerInput() . . . 11

2.3.3.2 logEvent() . . . 12

2.3.3.3 pollGameEvent() . . . 12

2.3.3.4 processEvents() . . . 12

2.3.3.5 processGameEvents() . . . 13

2.3.3.6 processSfmlEvents() . . . 13

2.3.3.7 render() . . . 13

2.3.3.8 renderBoundaries() . . . 13

2.3.3.9 run() . . . 13

2.3.3.10 update() . . . 14

2.4 tetris::Event::Location Struct Reference . . . 14

2.5 tetris::Brick::Position Struct Reference . . . 14

Index 15

Chapter 1

Class Index

1.1 Class List

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

tetris::Brick
Designed to initialize and render brick sprites . . . 3
tetris::Event
Designed to create and throw game events . . . 7
tetris::Game
Designed to initialize and to run the game . . . 10
tetris::Event::Location . . . 14
tetris::Brick::Position . . . 14

Chapter 2

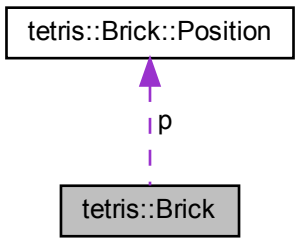
Class Documentation

2.1 tetris::Brick Class Reference

The Brick class is designed to initialize and render brick sprites.

`#include <brick.h>`

Collaboration diagram for tetris::Brick:



Classes

- struct Position

Public Member Functions

- Brick (const Vector2f &offset)

  Default constructor.
- void rotate ()

  Parameterized constructor.
- void move (const Vector2f &offset)

  moves the brick along a geometrical vector.
- void render (RenderWindow ∗window)

  renders the brick as a sprite using the OpenGL drawing services of SFML.
- bool inbound (Vector2u area)

  determines if the brick position is inside a particular area which dimensions are given.
- void getPosition (Position ∗)

  returns the brick's current position

Public Attributes

- struct tetris::Brick::Position p [4]

  position of each of the square blocks of the brick

Static Public Attributes

- static const String I_SHAPE

  I-shape brick –String constant.
- static const String Z_SHAPE

  Z-shape brick –String constant.
- static const String S_SHAPE

  S-shape brick –String constant.
- static const String T_SHAPE

  T-shape brick –String constant.
- static const String L_SHAPE

  L-shape brick –String constant.
- static const String J_SHAPE

  J-shape brick –String constant.
- static const String O_SHAPE

  O-shape brick –String constant.
- static const unsigned int block_size = 30

  Default block size for bricks in pixels.
- static const unsigned int wall_size = 15

  Default block size for bricks in pixels.

2.1.1 Detailed Description

The Brick class is designed to initialize and render brick sprites.

This class represents the brick objects represented on the screen. Bricks come in different shapes:

- I_SHAPE
- Z_SHAPE
- S_SHAPE
- T_SHAPE
- L_SHAPE
- J_SHAPE
- O_SHAPE.

Brick class provides the following services

- rotate – rotates the brick at 90°
- move – moves the brick along a 2D geometrical vector
- render – renders the brick on the Window provided.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Brick()

```
tetris::Brick::Brick (
            const Vector2f & offset )
```

Default constructor.

This constructor creates the brick object

Parameters

offset	This is the default location of the Brick on the stage

2.1.3 Member Function Documentation

2.1.3.1 getPosition()

```
void tetris::Brick::getPosition (
            Position *  )
```

returns the brick's current position

This functions returns the brick position on the stage.

Parameters

position	a structure holding the positions of all the blocks that are part of the brick

2.1.3.2 inbound()

```
bool tetris::Brick::inbound (
            Vector2u area )
```

determines if the brick position is inside a particular area which dimensions are given.

Use this function to limit the moves of a brick inside the boundaries of a particular area of the Game's window.

Parameters

area	area where the brick position will be verified if it sits in between

Returns

TRUE if the brick position is inside the boudaries of the area and FALSE if it's not

2.1.3.3 move()

```
void tetris::Brick::move (
            const Vector2f & offset )
```

moves the brick along a geometrical vector.

Use this function to move the bricks up, down, left, right. Vector2f Vector2u are classes provided by SFML framework for 2D transformation. The moves of the brick are limited by the boundaries of the area provided.

Parameters

offset	SFML Vector specifying the directions of the move.
area	SFML Vector specifying the boundaries of the move.

2.1.3.4 render()

```
void tetris::Brick::render (
            RenderWindow * window )
```

renders the brick as a sprite using the OpenGL drawing services of SFML.

Use this function to draw bricks with simple graphics without textures.

Parameters

window	address of the SFML window where the brick will render its sprite.

2.1.3.5 rotate()

```
void tetris::Brick::rotate ( )
```

Parameterized constructor.

This constructor creates the brick object that correspond to the shape provided as parameter.

Parameters

shape	name of one of the shapes bricks can take in the game.

rotates the brick counterclockwise.

Use this function as a handler of one of the key events in the game. the shape is rotated at at 90°.

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

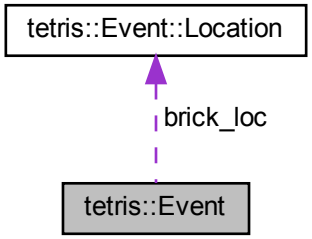
- include/brick.h

2.2 tetris::Event Class Reference

The Event class is designed to create and throw game events.

`#include <event.h>`

Collaboration diagram for tetris::Event:



Classes

- struct Location

Public Types

- enum EventType {
  RENDER, NEW_BRICK_DROP, BRICK_MOVING_UP, BRICK_STOP_MOVING_UP,
  BRICK_MOVING_DOWN, BRICK_STOP_MOVING_DOWN, BRICK_MOVING_LEFT, BRICK_STOP_MOVING_LEFT,
  BRICK_MOVING_RIGHT, BRICK_STOP_MOVING_RIGHT, BRICK_ROTATE, BRICK_STOP_ROTATE,
  BRICK_HIT_WALL, BRICK_OUT_OF_BOUNDS, ROW_COMPLETED, NO_SPACE_AVAILABLE,
  GAME_STARTED, GAME_OVER, GAME_QUIT, UNDEFINED }

  Enumeration of the different types of events.

Public Member Functions

- Event ()

  Default constructor.
- Event (Brick brick, Time t)

  Parameterized constructor which logs the position of the brick.
- Event (Brick brick, EventType eType, Time t)

  Parameterized constructor which logs the position of the brick.

Public Attributes

- EventType type

  Type of the event.
- Time time

  Time at which the event was reported.
- struct tetris::Event::Location brick_loc [4]

  position of each of the square blocks of the brick at the time of the event initialization

Friends

- std::ostream & **operator**<< (std::ostream &outputStream, const Event &e)

2.2.1 Detailed Description

The Event class is designed to create and throw game events.

This class represents the Event that may happen while the game is being played

- NEW_BRICK_DROP – this event is thrown every time a new brick is dropped on the stage
- BRICK_MOVED – this event is thrown every time the brick moves on the stage
- BRICK_ROTATE – this event is thrown every time the brick rotates on the stage
- BRICK_HIT_WALL – this event is thrown every time the brick hits the wall or the bottom of the stage
- BRICK_OUT_OF_BOUNDS – this event is thrown every time the brick position accidently appears to be out of bound
- ROW_COMPLETED – this event is thrown every time a completed row is detected by the game engine
- NO_SPACE_AVAILABLE – this event is thrown every time there's no more area on the stage available for a new brick drop
- GAME_OVER – this event is thrown when the game engines stops because there's no more space available

2.2.2 Member Enumeration Documentation

2.2.2.1 EventType

`enum tetris::Event::EventType`

Enumeration of the different types of events.

Enumerator

RENDER	this event is thrown when it is time for SFML to render the frame on the Window
NEW_BRICK_DROP	this event is thrown every time a new brick is dropped on the stage
BRICK_MOVING_UP	this event is thrown every time the brick moves up on the stage
BRICK_STOP_MOVING_UP	this event is thrown every time the brick stops moving up on the stage
BRICK_MOVING_DOWN	this event is thrown every time the brick moves down on the stage
BRICK_STOP_MOVING_DOWN	this event is thrown every time the brick stops moving down on the stage
BRICK_MOVING_LEFT	this event is thrown every time the brick moves left on the stage
BRICK_STOP_MOVING_LEFT	this event is thrown every time the brick stops moving left on the stage
BRICK_MOVING_RIGHT	this event is thrown every time the brick moves right on the stage
BRICK_STOP_MOVING_RIGHT	this event is thrown every time the brick stops moving right on the stage
BRICK_ROTATE	this event is thrown every time the brick rotates on the stage
BRICK_STOP_ROTATE	this event is thrown every time the brick stops rotating on the stage
BRICK_HIT_WALL	this event is thrown every time the brick hits the wall or the bottom of the stage
BRICK_OUT_OF_BOUNDS	this event is thrown every time the brick position accidently appears to be out of bound
ROW_COMPLETED	this event is thrown every time a completed row is detected by the game engine
NO_SPACE_AVAILABLE	this event is thrown every time there's no more area on the stage available for a new brick drop
GAME_STARTED	this event is thrown every time the game engine is started
GAME_OVER	this event is thrown when the game engines stops because there's no more space available
GAME_QUIT	this event is thrown when the game is stopped and the window is closed by the player
UNDEFINED	this is the default value when an event object is just created but undefined

2.2.3 Constructor & Destructor Documentation

2.2.3.1 Event() [1/3]

```
tetris::Event::Event ( )
```

Default constructor.

This constructor creates the event object. the eventType property is set to UNDEFINED.

2.2.3.2 Event() [2/3]

```
tetris::Event::Event (
            Brick brick,
            Time t )
```

Parameterized constructor which logs the position of the brick.

This constructor creates the event object. the eventType property is set to UNDEFINED. The position of the Brick at the time of the creation of the event is copied into the event.

Parameters

brick	The brick that is manipulated in the game.
t	The time at which the event is reported

2.2.3.3 Event() [3/3]

```
tetris::Event::Event (
            Brick brick,
            EventType eType,
            Time t )
```

Parameterized constructor which logs the position of the brick.

This constructor creates the event object. the eventType property is set to UNDEFINED. The position of the Brick at the time of the creation of the event is copied into the event.

Parameters

brick	The brick that is manipulated in the game.
eType	The type of event that is reported.
t	The time at which the event is reported.

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

- include/event.h

2.3 tetris::Game Class Reference

The Game class is designed to initialize and to run the game.

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

Public Member Functions

- Game ()

  Default constructor.
- void run ()

  runs the Game in an OpenGL window.

Static Public Attributes

- static const Time **TimePerFrame**
- static float PlayerSpeed

  The amount of time the Game Engine has to wait before refreshing the Window.

Protected Member Functions

- void processEvents ()

  Handles all events of the Game.
- void processSfmlEvents ()

  Handles events captured by SFML Engime.
- void processGameEvents ()

  Handles events provided to the Game Engine.
- void **processBrickEvents** ()
- void update (Time)

  Updates game data after each event processing loop.
- void render ()

  renders all sprites on the screen after each update.
- void renderBoundaries ()

  renders the boundaries of the stage
- void handlePlayerInput (Keyboard::Key key, bool isPressed, Time time)

  calls keyboard event handlers according to the key received.
- bool pollGameEvent (Event &event)

  pops any event in the event queue and returns it.
- void logEvent (Event event)

  prints a message explaining the event to the console.

2.3.1 Detailed Description

The Game class is designed to initialize and to run the game.

This class represents the Game Engine and provides the default service and functionalities for running the Game.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Game()

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

Default constructor.

This constructor creates the game object and initialize its default settings.

2.3.3 Member Function Documentation

2.3.3.1 handlePlayerInput()

```
void tetris::Game::handlePlayerInput (
            Keyboard::Key key,
            bool isPressed,
            Time time )  [protected]
```

calls keyboard event handlers according to the key received.

Use this function to define handlers for each of the keys you would like the game to handle.

Parameters

key	SFML key code e.g. Keyboard::A designate keyboard "A" key value
isPressed	if TRUE the key is pressed else the key is not pressed

2.3.3.2 logEvent()

```
void tetris::Game::logEvent (
            Event event )  [protected]
```

prints a message explaining the event to the console.

Use this function to track how the game handles events. This function will print events message and the brick position on the console.

Parameters

event	Game event that would be explained in the console.

2.3.3.3 pollGameEvent()

```
bool tetris::Game::pollGameEvent (
            Event & event )  [protected]
```

pops any event in the event queue and returns it.

This function is not blocking. If no event is found in the event queue it will return FALSE and the event parameter will return undefined. Otherwise it will return TRUE

Parameters

event	Game event that would be returned

Returns

TRUE if an event was returned and FALSE if the event queue is empty.

2.3.3.4 processEvents()

```
void tetris::Game::processEvents ( )  [protected]
```

Handles all events of the Game.

Use this function to call all event handlers.

2.3.3.5 processGameEvents()

```
void tetris::Game::processGameEvents ( )  [protected]
```

Handles events provided to the Game Engine.

Use this function to call different event handlers. Game Events are generated by the SFML handlers or by the Game itself. They can trigger the execution of some Game Rules. You should not call this function directly. You should instead call the processEvents() function.

2.3.3.6 processSfmlEvents()

```
void tetris::Game::processSfmlEvents ( )  [protected]
```

Handles events captured by SFML Engime.

This function call different graphic events' handlers. Graphic Events are captured by the Game Window. They cay can be keyboard events, time events, gamePad events. The handling of these events can generate game events. You should not call this function directly. You should instead call the processEvents() function.

2.3.3.7 render()

```
void tetris::Game::render ( )  [protected]
```

renders all sprites on the screen after each update.

Use this function in a loop in order to clear the window and redraw all game sprites with their new data.

2.3.3.8 renderBoundaries()

```
void tetris::Game::renderBoundaries ( )  [protected]
```

renders the boundaries of the stage

Use this function to draw boundaries of the game's stage sprites move would not overflow these boundaries

2.3.3.9 run()

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

runs the Game in an OpenGL window.

Use this function to run the Game after creating the Game object with the constructor.

2.3.3.10 update()

```
void tetris::Game::update (
            Time  )  [protected]
```

Updates game data after each event processing loop.

Use this function to update sprites position, game scores and objects data in the background.

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

- include/game.h

2.4 tetris::Event::Location Struct Reference

Public Attributes

- short **x**
- short **y**

The documentation for this struct was generated from the following file:

- include/event.h

2.5 tetris::Brick::Position Struct Reference

Public Attributes

- short **x**
- short **y**

The documentation for this struct was generated from the following file:

- include/brick.h

Index

Brick
    tetris::Brick, 5
BRICK_HIT_WALL
    tetris::Event, 9
BRICK_MOVING_DOWN
    tetris::Event, 9
BRICK_MOVING_LEFT
    tetris::Event, 9
BRICK_MOVING_RIGHT
    tetris::Event, 9
BRICK_MOVING_UP
    tetris::Event, 9
BRICK_OUT_OF_BOUNDS
    tetris::Event, 9
BRICK_ROTATE
    tetris::Event, 9
BRICK_STOP_MOVING_DOWN
    tetris::Event, 9
BRICK_STOP_MOVING_LEFT
    tetris::Event, 9
BRICK_STOP_MOVING_RIGHT
    tetris::Event, 9
BRICK_STOP_MOVING_UP
    tetris::Event, 9
BRICK_STOP_ROTATE
    tetris::Event, 9

Event
    tetris::Event, 9, 10
EventType
    tetris::Event, 8

Game
    tetris::Game, 11
GAME_OVER
    tetris::Event, 9
GAME_QUIT
    tetris::Event, 9
GAME_STARTED
    tetris::Event, 9
getPosition
    tetris::Brick, 5

handlePlayerInput
    tetris::Game, 11

inbound
    tetris::Brick, 5

logEvent
    tetris::Game, 12

move
    tetris::Brick, 6

NEW_BRICK_DROP
    tetris::Event, 9
NO_SPACE_AVAILABLE
    tetris::Event, 9

pollGameEvent
    tetris::Game, 12
processEvents
    tetris::Game, 12
processGameEvents
    tetris::Game, 12
processSfmlEvents
    tetris::Game, 13

RENDER
    tetris::Event, 9
render
    tetris::Brick, 6
    tetris::Game, 13
renderBoundaries
    tetris::Game, 13
rotate
    tetris::Brick, 6
ROW_COMPLETED
    tetris::Event, 9
run
    tetris::Game, 13

tetris::Brick, 3
    Brick, 5
    getPosition, 5
    inbound, 5
    move, 6
    render, 6
    rotate, 6
tetris::Brick::Position, 14
tetris::Event, 7
    BRICK_HIT_WALL, 9
    BRICK_MOVING_DOWN, 9
    BRICK_MOVING_LEFT, 9
    BRICK_MOVING_RIGHT, 9
    BRICK_MOVING_UP, 9
    BRICK_OUT_OF_BOUNDS, 9
    BRICK_ROTATE, 9
    BRICK_STOP_MOVING_DOWN, 9
    BRICK_STOP_MOVING_LEFT, 9
    BRICK_STOP_MOVING_RIGHT, 9

BRICK_STOP_MOVING_UP, 9
BRICK_STOP_ROTATE, 9
Event, 9, 10
EventType, 8
GAME_OVER, 9
GAME_QUIT, 9
GAME_STARTED, 9
NEW_BRICK_DROP, 9
NO_SPACE_AVAILABLE, 9
RENDER, 9
ROW_COMPLETED, 9
UNDEFINED, 9
tetris::Event::Location, 14
tetris::Game, 10
Game, 11
handlePlayerInput, 11
logEvent, 12
pollGameEvent, 12
processEvents, 12
processGameEvents, 12
processSfmlEvents, 13
render, 13
renderBoundaries, 13
run, 13
update, 13

UNDEFINED
tetris::Event, 9
update
tetris::Game, 13