Tedm

1.0

Generated by Doxygen 1.8.6

Fri Apr 28 2017 04:58:35

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 Ball Class Reference 7

4.1.1 Detailed Description 7

4.1.2 Constructor & Destructor Documentation 8

4.1.2.1 Ball 8

4.1.3 Member Function Documentation 8

4.1.3.1 get_x 8

4.1.3.2 get_y 8

4.1.3.3 update_trajectory 8

4.2 Tedm::Context Class Reference 8

4.2.1 Detailed Description 9

4.2.2 Constructor & Destructor Documentation 9

4.2.2.1 Context 9

4.3 Tedm::Event Class Reference 9

4.3.1 Detailed Description 10

4.3.2 Constructor & Destructor Documentation 10

4.3.2.1 Event 10

4.3.3 Member Function Documentation 10

4.3.3.1 currentMouseButtonPress 10

4.3.3.2 currentMouseButtonX 10

4.3.3.3 currentMouseButtonY 11

4.3.3.4 getEvent 11

4.3.3.5 getKeySymbol 11

4.3.3.6 getMouseX . . . 11
4.3.3.7 getMouseY . . . 11
4.3.3.8 getType . . . 11
4.3.3.9 leftButtonPress . . . 11
4.3.3.10 middleButtonPress . . . 12
4.3.3.11 poll . . . 12
4.3.3.12 rightButtonPress . . . 12
4.4 Tedm::EventHandler Class Reference . . . 12
4.4.1 Member Function Documentation . . . 14
4.4.1.1 addExitListener . . . 14
4.4.1.2 addKeyDownListener . . . 14
4.4.1.3 addKeyUpListener . . . 14
4.4.1.4 addLButtonDownListener . . . 14
4.4.1.5 addLButtonUpListener . . . 15
4.4.1.6 addMButtonDownListener . . . 15
4.4.1.7 addMButtonUpListener . . . 15
4.4.1.8 addMouseMoveListener . . . 15
4.4.1.9 addMouseWheelListener . . . 15
4.4.1.10 addRButtonDownListener . . . 15
4.4.1.11 addRButtonUpListener . . . 16
4.4.1.12 addTrigger . . . 16
4.4.1.13 addUserListener . . . 16
4.4.1.14 removeExitListener . . . 16
4.4.1.15 removeKeyDownListener . . . 16
4.4.1.16 removeKeyUpListener . . . 16
4.4.1.17 removeLButtonDownListener . . . 17
4.4.1.18 removeLButtonUpListener . . . 17
4.4.1.19 removeMButtonDownListener . . . 17
4.4.1.20 removeMButtonUpListener . . . 17
4.4.1.21 removeMouseMoveListener . . . 17
4.4.1.22 removeMouseWheelListener . . . 17
4.4.1.23 removeRButtonDownListener . . . 18
4.4.1.24 removeRButtonUpListener . . . 18
4.4.1.25 removeTrigger . . . 18
4.4.1.26 removeUserListener . . . 18
4.5 Tedm::EventListener Class Reference . . . 18
4.5.1 Detailed Description . . . 19
4.6 Tedm::EventTrigger Class Reference . . . 19
4.7 Tedm::Game Class Reference . . . 19
4.7.1 Detailed Description . . . 20

4.7.2 Member Function Documentation . . . 20
4.7.2.1 setStartState . . . 20
4.7.2.2 setTargetFramerate . . . 21
4.7.2.3 setWindowTitle . . . 21
4.7.2.4 transition . . . 21
4.8 GameState Class Reference . . . 21
4.9 Tedm::Graphics Class Reference . . . 23
4.9.1 Member Function Documentation . . . 23
4.9.1.1 add_background . . . 23
4.9.1.2 draw . . . 23
4.9.1.3 draw . . . 23
4.9.1.4 init . . . 24
4.9.1.5 isInitialized . . . 25
4.9.1.6 loadIMG . . . 25
4.9.1.7 loadTexture . . . 25
4.9.1.8 setWindowTitle . . . 25
4.10 Tedm::KeyEventListener Class Reference . . . 26
4.10.1 Detailed Description . . . 26
4.11 Tedm::Logger Class Reference . . . 26
4.11.1 Detailed Description . . . 27
4.11.2 Member Function Documentation . . . 27
4.11.2.1 getLevel . . . 27
4.11.2.2 log . . . 27
4.11.2.3 log_debug . . . 27
4.11.2.4 log_error . . . 27
4.11.2.5 log_info . . . 28
4.11.2.6 log_warning . . . 28
4.11.2.7 setLevel . . . 28
4.11.3 Member Data Documentation . . . 28
4.11.3.1 level_strings . . . 28
4.12 Tedm::MouseButtonListener Class Reference . . . 28
4.12.1 Detailed Description . . . 29
4.13 Tedm::MouseMoveListener Class Reference . . . 29
4.13.1 Detailed Description . . . 29
4.14 Tedm::MouseWheelListener Class Reference . . . 29
4.14.1 Detailed Description . . . 29
4.15 Tedm::Object Class Reference . . . 29
4.15.1 Detailed Description . . . 31
4.15.2 Constructor & Destructor Documentation . . . 31
4.15.2.1 Object . . . 31

4.15.2.2 Object . . . 31
4.15.2.3 Object . . . 31
4.15.2.4 Object . . . 31
4.15.3 Member Function Documentation . . . 32
4.15.3.1 collision . . . 32
4.15.3.2 get_height . . . 32
4.15.3.3 get_width . . . 32
4.15.3.4 get_x . . . 32
4.15.3.5 get_y . . . 32
4.15.3.6 set_position . . . 33
4.15.3.7 set_sprite . . . 34
4.15.3.8 set_x . . . 34
4.15.3.9 set_y . . . 34
4.16 Player Class Reference . . . 34
4.16.1 Detailed Description . . . 35
4.16.2 Constructor & Destructor Documentation . . . 35
4.16.2.1 Player . . . 35
4.16.3 Member Function Documentation . . . 35
4.16.3.1 get_height . . . 35
4.16.3.2 get_y . . . 35
4.16.3.3 set_pos . . . 35
4.17 Tedm::Player_base Class Reference . . . 36
4.17.1 Detailed Description . . . 36
4.17.2 Constructor & Destructor Documentation . . . 36
4.17.2.1 Player_base . . . 36
4.17.2.2 Player_base . . . 36
4.18 Player_KeyBoard_Listener Class Reference . . . 37
4.18.1 Detailed Description . . . 37
4.19 Pong_State Class Reference . . . 37
4.19.1 Detailed Description . . . 38
4.19.2 Constructor & Destructor Documentation . . . 38
4.19.2.1 Pong_State . . . 38
4.19.3 Member Function Documentation . . . 38
4.19.3.1 init . . . 38
4.20 Quit_Listener Class Reference . . . 39
4.21 Tedm::Sprite_base Class Reference . . . 39
4.21.1 Detailed Description . . . 40
4.21.2 Constructor & Destructor Documentation . . . 40
4.21.2.1 Sprite_base . . . 40
4.21.2.2 Sprite_base . . . 40

4.21.2.3 Sprite_base . . . 40
4.21.3 Member Function Documentation . . . 40
4.21.3.1 get_pos . . . 40
4.21.3.2 get_sprite . . . 41
4.21.3.3 get_src . . . 41
4.21.3.4 set_height_width . . . 41
4.21.3.5 set_position . . . 41
4.21.3.6 set_source_pos . . . 41
4.21.3.7 set_sprite . . . 41
4.22 Tedm::State Class Reference . . . 42
4.22.1 Constructor & Destructor Documentation . . . 42
4.22.1.1 State . . . 42
4.22.2 Member Function Documentation . . . 43
4.22.2.1 getID . . . 43
4.23 Tedm::Texture Class Reference . . . 43
4.23.1 Detailed Description . . . 43
4.24 Timer Class Reference . . . 43
4.24.1 Detailed Description . . . 44
4.24.2 Member Function Documentation . . . 44
4.24.2.1 get_ticks . . . 44
4.24.2.2 is_paused . . . 44
4.24.2.3 is_started . . . 44
4.25 Tedm::UserListener Class Reference . . . 44
4.25.1 Detailed Description . . . 45

5 File Documentation **47**

5.1 demos/pong.cpp File Reference . . . 47
5.1.1 Detailed Description . . . 47
5.2 src/Context.cpp File Reference . . . 48
5.2.1 Detailed Description . . . 48
5.3 src/Context.h File Reference . . . 48
5.3.1 Detailed Description . . . 48
5.4 src/events/Event.cpp File Reference . . . 49
5.4.1 Detailed Description . . . 49
5.5 src/events/Event.h File Reference . . . 49
5.5.1 Detailed Description . . . 49
5.6 src/events/EventHandler.cpp File Reference . . . 50
5.6.1 Detailed Description . . . 50
5.7 src/events/EventHandler.h File Reference . . . 50
5.7.1 Detailed Description . . . 50

5.8 src/events/EventListener.h File Reference . . . 51
5.8.1 Detailed Description . . . 51
5.9 src/Game.cpp File Reference . . . 52
5.9.1 Detailed Description . . . 52
5.10 src/Game.h File Reference . . . 52
5.10.1 Detailed Description . . . 52
5.11 src/Graphics.cpp File Reference . . . 53
5.11.1 Detailed Description . . . 53
5.12 src/Graphics.h File Reference . . . 53
5.12.1 Detailed Description . . . 53
5.13 src/img/Texture.h File Reference . . . 54
5.13.1 Detailed Description . . . 54
5.14 src/objects/player.h File Reference . . . 54
5.14.1 Detailed Description . . . 55
5.15 src/objects/sprite.h File Reference . . . 55
5.15.1 Detailed Description . . . 55
5.16 src/State.h File Reference . . . 55
5.16.1 Detailed Description . . . 56
5.16.2 DESCRIPTION . . . 56
5.17 src/utils/Timer.cpp File Reference . . . 56
5.17.1 Detailed Description . . . 56
5.18 src/utils/Timer.h File Reference . . . 57
5.18.1 Detailed Description . . . 57

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Tedm::Context . . . 8
Tedm::Event . . . 9
Tedm::EventHandler . . . 12
Tedm::EventListener . . . 18
  Quit_Listener . . . 39
Tedm::EventTrigger . . . 19
Tedm::Game . . . 19
Tedm::Graphics . . . 23
Tedm::KeyEventListener . . . 26
  Player_KeyBoard_Listener . . . 37
Tedm::Logger . . . 26
Tedm::MouseButtonListener . . . 28
Tedm::MouseMoveListener . . . 29
Tedm::MouseWheelListener . . . 29
Tedm::Object . . . 29
  Ball . . . 7
  Tedm::Player_base . . . 36
    Player . . . 34
Tedm::Sprite_base . . . 39
Tedm::State . . . 42
  GameState . . . 21
  GameState . . . 21
  GameState . . . 21
  Pong_State . . . 37
Tedm::Texture . . . 43
Timer . . . 43
Tedm::UserListener . . . 44

Chapter 2

Class Index

2.1 Class List

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

Ball
: User makes a class for any object type in the game. In this case the ball is the only non-player object . . . 7

Tedm::Context
: Contains details about the game condition . . . 8

Tedm::Event
: Create hooks for key presses and other in-game events that trigger changes in game state . . . 9

Tedm::EventHandler . . . 12

Tedm::EventListener
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is the default case for custom events that are not SDL key events 18

Tedm::EventTrigger . . . 19

Tedm::Game
: Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game . . . 19

GameState . . . 21

Tedm::Graphics . . . 23

Tedm::KeyEventListener
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon keypress . . . 26

Tedm::Logger
: Logger for debugging . . . 26

Tedm::MouseButtonListener
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse button press . . . 28

Tedm::MouseMoveListener
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse movement . . . 29

Tedm::MouseWheelListener
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse wheel interaction . . . 29

Tedm::Object
: Basic game element. Any item in game should inherit object . . . 29

Player
: User makes a Player class which inherits Player_base and defines functionality specific to their game. This includes default size, position, functions for user input, etc . . . 34

Tedm::Player_base
Define player object for game interaction The developer can inherit player to create custom players for each game . . . 36
Player_KeyBoard_Listener
The user greates a KeyEventListener class to define the functions that will execute in response to user input, inheriting the KeyEventListener class and overriding the appropriate virtual functions 37
Pong_State
The user creates State class for each game state which inherits the State class and implements elements specific to the state . . . 37
Quit_Listener . . . 39
Tedm::Sprite_base
Graphic representation of game element . . . 39
Tedm::State . . . 42
Tedm::Texture
Define Texture to store object or background image . . . 43
Timer
Timer for maintaining frame rate . . . 43
Tedm::UserListener
Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event . . . 44

Chapter 3

File Index

3.1 File List

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

demos/pong.cpp
: Simple example game demonstrating Tedm API . . . 47

src/Context.cpp
: Contains details about the game condition . . . 48

src/Context.h
: Contains details about the game condition . . . 48

src/Game.cpp
: Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game . . . 52

src/Game.h
: Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game . . . 52

src/Graphics.cpp
: The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects . . . 53

src/Graphics.h
: The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects . . . 53

src/State.h
: State object used for defining a game state . . . 55

src/events/Event.cpp
: Create hooks for key presses and other in-game events that trigger changes in game state . . . 49

src/events/Event.h
: Create hooks for key presses and other in-game events that trigger changes in game state . . . 49

src/events/EventHandler.cpp
: Create handlers for Events so developers can indicate which events their game will respond to 50

src/events/EventHandler.h
: Create handlers for Events so developers can indicate which events their game will respond to 50

src/events/EventListener.h
: Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event . . . 51

src/events/**EventTrigger.h** . . . **??**

src/img/Texture.h
: Define Texture to store object or background image . . . 54

src/objects/**object.h** . . . **??**

src/objects/player.h
Basic game element. Any item in game should inherit object 54
src/objects/sprite.h
Graphic representation of game element 55
src/utils/**Logger.h** . . . . . **??**
src/utils/Timer.cpp
Timer for maintaining frame rate 56
src/utils/Timer.h
Timer for maintaining frame rate 57

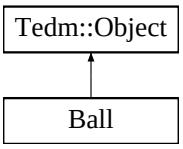
Chapter 4

Class Documentation

4.1 Ball Class Reference

the user makes a class for any object type in the game. In this case the ball is the only non-player object

Inheritance diagram for Ball:



Public Member Functions

- Ball (Graphics &g, std::string filename, int posx, int posy, int srcx, int srcy)

  The Ball constructor sets the height, weight, position, and position of the sprite within the image.
- void reset ()

  Return the ball to starting position and set the speed to default(1)
- void update_trajectory (Player &p)

  Change the trajectory of ball based on the position of contact with the paddle.
- void update_trajectory ()

  Reverse trajectory upon direct contact.
- void update_pos ()

  Move the ball each frame based on velocity.
- void set_sprite (string filename)

  Set the image to represent the Ball The file path to the image.
- int get_x ()

  Get the x coordinate position.
- int get_y ()

  Get the y coordinate position.

Additional Inherited Members

4.1.1 Detailed Description

the user makes a class for any object type in the game. In this case the ball is the only non-player object

See Also

Object

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Ball::Ball (Graphics & *g*, std::string *filename*, int *posx*, int *posy*, int *srcx*, int *srcy*) [inline]

The Ball constructor sets the height, weight, position, and position of the sprite within the image.

Parameters

posx	the x coordinate
posy	the y coordinate
srcx	the x coordinate of the sprite within the image
srcy	the y coordinate of the sprite within the image

4.1.3 Member Function Documentation

4.1.3.1 int Ball::get_x ( ) [inline]

Get the x coordinate position.

Returns

the x coordinate position

4.1.3.2 int Ball::get_y ( ) [inline]

Get the y coordinate position.

Returns

the y coordinate position

4.1.3.3 void Ball::update_trajectory (Player & *p*) [inline]

Change the trajectory of ball based on the position of contact with the paddle.

Parameters

p	the paddle which made contact

See Also

https://gamedev.stackexchange.com/questions/4253/in-pong-how-do-you-calculate-the-balls-direction-when-it-bounces-off-the-paddl

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

- demos/pong.cpp

4.2 Tedm::Context Class Reference

Contains details about the game condition.

`#include <Context.h>`

Public Member Functions

- Context ()

  Constructor sets the default starting conditions with default screen size.
- Context (int w, int h)

  Constructor sets the default starting conditions with user-defined screen size.

Public Attributes

- int **width**
- int **height**
- std::string **windowTitle**
- bool **isRunning**
- int **timeSinceLastLoop**
- long **targetFramerate**
- bool **isPaused**

4.2.1 Detailed Description

Contains details about the game condition.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Tedm::Context::Context (int *w*, int *h*) [inline]

Constructor sets the default starting conditions with user-defined screen size.

Parameters

w	Screen width
h	Screen height

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

- src/Context.h

4.3 Tedm::Event Class Reference

Create hooks for key presses and other in-game events that trigger changes in game state.

`#include <Event.h>`

Public Member Functions

- Event ()

  Constructor.
- SDL_Event & getEvent ()

  Retrieve the SDL_Event from the event object.
- int getType ()

  Retrieve the type of SDL_Event.
- SDL_Keycode getKeySymbol ()

  Retrieve the keycode for the key event.

- int getMouseX ()
  Retrieve the mouse x coordinate movement.
- int getMouseY ()
  Retrieve the mouse y coordinate movement.
- int **getMouseXRel** ()
- int **getMouseYRel** ()
- bool leftButtonPress ()
  Retrieve the mouse left button press event.
- bool rightButtonPress ()
  Retrieve the mouse right button press event.
- bool middleButtonPress ()
  Retrieve the mouse middle button press event.
- int currentMouseButtonPress ()
  Retrieve the mouse button pressed.
- int currentMouseButtonX ()
  Retrieve the mouse X button press event.
- int currentMouseButtonY ()
  Retrieve the mouse Y button press event.
- bool poll ()
  check if a SDL_Event has occurred such as a keypress or mouse event

4.3.1 Detailed Description

Create hooks for key presses and other in-game events that trigger changes in game state.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Tedm::Event::Event ( ) [inline]

Constructor.

Parameters

e	the SDL_Event to hook

4.3.3 Member Function Documentation

4.3.3.1 int Tedm::Event::currentMouseButtonPress ( ) [inline]

Retrieve the mouse button pressed.

Returns

the button keycode of the button pressed

4.3.3.2 int Tedm::Event::currentMouseButtonX ( ) [inline]

Retrieve the mouse X button press event.

Returns

the movement data during the press event

4.3.3.3 int Tedm::Event::currentMouseButtonY ( ) [inline]

Retrieve the mouse Y button press event.

Returns

the movement data during the press event

4.3.3.4 SDL_Event& Tedm::Event::getEvent ( ) [inline]

Retrieve the SDL_Event from the event object.

Returns

The event

4.3.3.5 SDL_Keycode Tedm::Event::getKeySymbol ( ) [inline]

Retrieve the keycode for the key event.

Returns

The keycode for the key event

4.3.3.6 int Tedm::Event::getMouseX ( ) [inline]

Retrieve the mouse x coordinate movement.

Returns

The x coordinate movement distance

4.3.3.7 int Tedm::Event::getMouseY ( ) [inline]

Retrieve the mouse y coordinate movement.

Returns

The y coordinate movement distance

4.3.3.8 int Tedm::Event::getType ( ) [inline]

Retrieve the type of SDL_Event.

Returns

The event type

4.3.3.9 bool Tedm::Event::leftButtonPress ( ) [inline]

Retrieve the mouse left button press event.

Returns

the movement data during the press event

4.3.3.10 bool Tedm::Event::middleButtonPress ( ) [inline]

Retrieve the mouse middle button press event.

Returns

the movement data during the press event

4.3.3.11 bool Tedm::Event::poll ( ) [inline]

check if a SDL_Event has occurred such as a keypress or mouse event

Returns

True if an Event has occurred

4.3.3.12 bool Tedm::Event::rightButtonPress ( ) [inline]

Retrieve the mouse right button press event.

Returns

the movement data during the press event

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

- src/events/Event.h

4.4 Tedm::EventHandler Class Reference

Public Member Functions

- void addKeyDownListener (std::shared_ptr< KeyEventListener > eventListener)
  Add listener for keydown event.
- void addKeyUpListener (std::shared_ptr< KeyEventListener > eventListener)
  Add listener for keyup event.
- void addMouseMoveListener (std::shared_ptr< MouseMoveListener > eventListener)
  Add listener for mouse move event.
- void addMouseWheelListener (std::shared_ptr< MouseWheelListener > eventListener)
  Add listener for mouse wheel event.
- void addLButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)
  Add listener for left mouse button down event.
- void addLButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)
  Add listener for left mouse button up event.
- void addRButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)
  Add listener for right mouse button down event.
- void addRButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)
  Add listener for right mouse button up event.
- void addMButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)
  Add listener for middle mouse button down event.
- void addMButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)

Add listener for middle mouse button up event.

- void addExitListener (std::shared_ptr< EventListener > eventListener)

  Add listener for exit event.
- void addUserListener (std::shared_ptr< UserListener > eventListener)

  Add listener for user event.
- void removeKeyDownListener (std::shared_ptr< KeyEventListener > eventListener)

  Remove listener for keydown event.
- void removeKeyUpListener (std::shared_ptr< KeyEventListener > eventListener)

  Remove listener for keyup event.
- void removeMouseMoveListener (std::shared_ptr< MouseMoveListener > eventListener)

  Remove listener for mouse move event.
- void removeMouseWheelListener (std::shared_ptr< MouseWheelListener > eventListener)

  Remove listener for mouse wheel event.
- void removeLButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for left mouse button down event.
- void removeLButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for left mouse button up event.
- void removeRButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for right mouse button down event.
- void removeRButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for right mouse button up event.
- void removeMButtonDownListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for middle mouse button down event.
- void removeMButtonUpListener (std::shared_ptr< MouseButtonListener > eventListener)

  Remove listener for right mouse button up event.
- void removeExitListener (std::shared_ptr< EventListener > eventListener)

  Remove listener for exit event.
- void removeUserListener (std::shared_ptr< UserListener > eventListener)

  Remove listener for user event.
- virtual void addTrigger (std::shared_ptr< EventTrigger > trigger)

  Add event trigger to be checked Event triggers are custom events which are not standard input.
- virtual void removeTrigger (std::shared_ptr< EventTrigger > trigger)

  Remove event trigger.
- void checkListeners ()

  check all registered event listeners
- void process ()

  When an event has occurred, check for listeners and process handlers.

Protected Attributes

- std::vector< std::shared_ptr < EventListener > > **_ExitEvents**
- std::vector< std::shared_ptr < UserListener > > **_UserEvents**
- std::vector< std::shared_ptr < MouseButtonListener > > **_LButtonDownEvents**
- std::vector< std::shared_ptr < MouseButtonListener > > **_LButtonUpEvents**
- std::vector< std::shared_ptr < MouseButtonListener > > **_RButtonDownEvents**
- std::vector< std::shared_ptr < MouseButtonListener > > **_RButtonUpEvents**

- std::vector< std::shared_ptr < MouseButtonListener > > **_MButtonDownEvents**
- std::vector< std::shared_ptr < MouseButtonListener > > **_MButtonUpEvents**
- std::vector< std::shared_ptr < MouseMoveListener > > **_MouseMoveEvents**
- std::vector< std::shared_ptr < MouseWheelListener > > **_MouseWheelEvents**
- std::vector< std::shared_ptr < KeyEventListener > > **_KeyDownEvents**
- std::vector< std::shared_ptr < KeyEventListener > > **_KeyUpEvents**
- std::vector< std::shared_ptr < EventTrigger > > **_eventTriggers**
- Event **event**

4.4.1 Member Function Documentation

4.4.1.1 void Tedm::EventHandler::addExitListener (std::shared_ptr< **EventListener** > *eventListener*) [inline]

Add listener for exit event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.2 void Tedm::EventHandler::addKeyDownListener (std::shared_ptr< **KeyEventListener** > *eventListener*) [inline]

Add listener for keydown event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.3 void Tedm::EventHandler::addKeyUpListener (std::shared_ptr< **KeyEventListener** > *eventListener*) [inline]

Add listener for keyup event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.4 void Tedm::EventHandler::addLButtonDownListener (std::shared_ptr< **MouseButtonListener** > *eventListener*) [inline]

Add listener for left mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.5 void Tedm::EventHandler::addLButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Add listener for left mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.6 void Tedm::EventHandler::addMButtonDownListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Add listener for middle mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.7 void Tedm::EventHandler::addMButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Add listener for middle mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.8 void Tedm::EventHandler::addMouseMoveListener (std::shared_ptr< MouseMoveListener > *eventListener*) [inline]

Add listener for mouse move event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.9 void Tedm::EventHandler::addMouseWheelListener (std::shared_ptr< MouseWheelListener > *eventListener*) [inline]

Add listener for mouse wheel event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.10 void Tedm::EventHandler::addRButtonDownListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Add listener for right mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.11 void Tedm::EventHandler::addRButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Add listener for right mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.12 void Tedm::EventHandler::addTrigger (std::shared_ptr< EventTrigger > *trigger*) [virtual]

Add event trigger to be checked Event triggers are custom events which are not standard input.

Parameters

trigger	the custom event trigger to be added

4.4.1.13 void Tedm::EventHandler::addUserListener (std::shared_ptr< UserListener > *eventListener*) [inline]

Add listener for user event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.14 void Tedm::EventHandler::removeExitListener (std::shared_ptr< EventListener > *eventListener*) [inline]

Remove listener for exit event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.15 void Tedm::EventHandler::removeKeyDownListener (std::shared_ptr< KeyEventListener > *eventListener*) [inline]

Remove listener for keydown event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.16 void Tedm::EventHandler::removeKeyUpListener (std::shared_ptr< KeyEventListener > *eventListener*) [inline]

Remove listener for keyup event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.17 void Tedm::EventHandler::removeLButtonDownListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for left mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.18 void Tedm::EventHandler::removeLButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for left mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.19 void Tedm::EventHandler::removeMButtonDownListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for middle mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.20 void Tedm::EventHandler::removeMButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for right mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.21 void Tedm::EventHandler::removeMouseMoveListener (std::shared_ptr< MouseMoveListener > *eventListener*) [inline]

Remove listener for mouse move event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.22 void Tedm::EventHandler::removeMouseWheelListener (std::shared_ptr< MouseWheelListener > *eventListener*) [inline]

Remove listener for mouse wheel event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.23 void Tedm::EventHandler::removeRButtonDownListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for right mouse button down event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.24 void Tedm::EventHandler::removeRButtonUpListener (std::shared_ptr< MouseButtonListener > *eventListener*) [inline]

Remove listener for right mouse button up event.

Parameters

eventListener	the listener which defines the action to occur upon event

4.4.1.25 void Tedm::EventHandler::removeTrigger (std::shared_ptr< EventTrigger > *trigger*) [virtual]

Remove event trigger.

Parameters

trigger	the item to remove

4.4.1.26 void Tedm::EventHandler::removeUserListener (std::shared_ptr< UserListener > *eventListener*) [inline]

Remove listener for user event.

Parameters

eventListener	the listener which defines the action to occur upon event

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

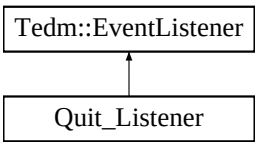
- src/events/EventHandler.h
- src/events/EventHandler.cpp

4.5 Tedm::EventListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is the default case for custom events that are not SDL key events.

`#include <EventListener.h>`

Inheritance diagram for Tedm::EventListener:



Public Member Functions

- virtual void **operator()** ()=0

4.5.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is the default case for custom events that are not SDL key events.

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

- src/events/EventListener.h

4.6 Tedm::EventTrigger Class Reference

Public Member Functions

- virtual bool **triggered** ()=0
- virtual void **operator()** ()=0

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

- src/events/EventTrigger.h

4.7 Tedm::Game Class Reference

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

`#include <Game.h>`

Public Member Functions

- Game ()

  Constructor creates a game, with context and state, at default screen size.
- Game (Context ctx)

  Constructor creates a game, with context and state.
- virtual ~Game ()

  Default destructor.
- void mainLoop ()

  main game function. Maintains frame rate and executes virtual update() function in child class. Checks for events and user input.
- void setTargetFramerate (int framerate)

  Sets the target frame rate.
- void setWindowTitle (std::string windowTitle)

  Sets the title of the game window.
- void shutdown ()

  Ends the game.
- void **registerState** (std::string id, std::shared_ptr< State > s)
- void transition (std::string newStateId)

  Transition to a new state.
- void setStartState (std::string startStateId)

  Set the start state of the Game.

Protected Member Functions

- virtual bool init ()

  initialize the game. Initialize graphics
- virtual void destroy ()

  Destroy and clean up the game.
- virtual void pause ()

  Pause execution.
- virtual void resume ()

  End pause condition.

Protected Attributes

- EventHandler **eventHandler**
- Graphics **graphics**
- Context **context**
- std::unordered_map < std::string, std::shared_ptr < State > > **state_id_dict**
- std::string **currentStateId**
- std::string **startStateId**
- Logger **log**
- std::string **nextStateId**
- bool **doTransition**

Friends

- class **State**

4.7.1 Detailed Description

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

4.7.2 Member Function Documentation

4.7.2.1 void Tedm::Game::setStartState (std::string *startStateId*)

Set the start state of the Game.

Parameters

startStateId	The string identifier of the start state

4.7.2.2 void Tedm::Game::setTargetFramerate (int *framerate*)

Sets the target frame rate.

Parameters

fps	the desired rate

4.7.2.3 void Tedm::Game::setWindowTitle (std::string *windowTitle*)

Sets the title of the game window.

Parameters

windowTitle	the new title

4.7.2.4 void Tedm::Game::transition (std::string *newStateId*)

Transition to a new state.

Parameters

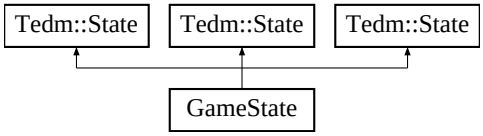
newStateId	the string identifier of the new state

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

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

4.8 GameState Class Reference

Inheritance diagram for GameState:



Public Member Functions

- **GameState** (Tedm::Game &g)
- bool init () override

  virtual function to be overridden by game-specific state Contains logic executed at game start
- void destroy () override

  virtual function to be overridden by game-specific state Cleans state listeners in preperation for new state
- void paused () override

  virtual function to be overridden by game-specific state Pauses the game
- void resumed () override

  virtual function to be overridden by game-specific state Un-pauses the game
- void update () override

  virtual function to be overridden by game-specific state Contains logic executed at each frame
- void render () override

  virtual function to be overridden by game-specific state Draws the state-appropriate items to the screen
- **GameState** (Tedm::Game &g)
- bool init () override

  virtual function to be overridden by game-specific state Contains logic executed at game start
- void destroy () override

  virtual function to be overridden by game-specific state Cleans state listeners in preperation for new state
- void paused () override

  virtual function to be overridden by game-specific state Pauses the game
- void resumed () override

  virtual function to be overridden by game-specific state Un-pauses the game
- void update () override

virtual function to be overridden by game-specific state Contains logic executed at each frame

- void render () override

  virtual function to be overridden by game-specific state Draws the state-appropriate items to the screen
- **GameState** (Tedm::Game &g)
- bool init () override

  virtual function to be overridden by game-specific state Contains logic executed at game start
- void destroy () override

  virtual function to be overridden by game-specific state Cleans state listeners in preperation for new state
- void paused () override

  virtual function to be overridden by game-specific state Pauses the game
- void resumed () override

  virtual function to be overridden by game-specific state Un-pauses the game
- void update () override

  virtual function to be overridden by game-specific state Contains logic executed at each frame
- void render () override

  virtual function to be overridden by game-specific state Draws the state-appropriate items to the screen

Public Attributes

- SDL_Texture * **background**
- Tedm::Object **blaster**

Additional Inherited Members

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

- demos/part1.cpp
- demos/part2.cpp
- demos/part3.cpp

4.9 Tedm::Graphics Class Reference

Public Member Functions

- Graphics ()

  Constructor waits for initialization.
- ~Graphics ()

  Default destructor.
- void destroy ()

  Destroy resources.
- bool init (int height, int width, std::string name)

  Initialize graphics.
- SDL_Texture * loadTexture (std::string path) const

  Load an image into a texture object. This is needed to render a background or a sprite.
- SDL_Surface * loadIMG (SDL_PixelFormat *format, std::string filename) const

  Load an image into an SDL_Surface object.
- SDL_Texture * add_background (std::string filename) const

  define the current background
- bool isInitialized ()

  Check if graphics are initialized.

- void setWindowTitle (std::string basic_string)

  Set the title of the game window.
- void draw (SDL_Texture *texture) const

  draw a texture onto the screen
- void draw (SDL_Texture *texture, SDL_Rect *src, SDL_Rect *tgt) const

  draw a sprite onto the screen
- void present () const

  render the screen

4.9.1 Member Function Documentation

4.9.1.1 SDL_Texture * Tedm::Graphics::add_background (std::string *filename*) const

define the current background

Parameters

filename	the file path

4.9.1.2 void Tedm::Graphics::draw (SDL_Texture * *texture*) const

draw a texture onto the screen

Parameters

texture	the item to draw

4.9.1.3 void Tedm::Graphics::draw (SDL_Texture * *texture,* SDL_Rect * *src,* SDL_Rect * *tgt*) const

draw a sprite onto the screen

texture the sprite image src the location of the image in the sprite tgt the location on the screen

4.9.1.4 bool Tedm::Graphics::init (int *height,* int *width,* std::string *name*)

Initialize graphics.

Parameters

height	the screen height
width	the screen width
the	name for the screen window

Returns

true on success

4.9.1.5 bool Tedm::Graphics::isInitialized ( )

Check if graphics are initialized.

Returns

true if graphics are initialized

4.9.1.6 SDL_Surface * Tedm::Graphics::loadIMG (SDL_PixelFormat * *format,* std::string *filename*) const

Load an image into an SDL_Surface object.

Parameters

format	the file format
filename	the file path

Returns

the new surface

4.9.1.7 SDL_Texture * Tedm::Graphics::loadTexture (std::string *path*) const

Load an image into a texture object. This is needed to render a background or a sprite.

Parameters

path	the file path

Returns

the new texture

4.9.1.8 void Tedm::Graphics::setWindowTitle (std::string *basic_string*)

Set the title of the game window.

Parameters

basic_string	the title

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

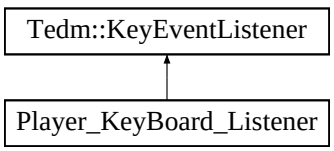
- src/Graphics.h
- src/Graphics.cpp

4.10 Tedm::KeyEventListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon keypress.

`#include <EventListener.h>`

Inheritance diagram for Tedm::KeyEventListener:



Public Member Functions

- virtual void **operator()** (SDL_Keycode sym)=0

4.10.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon keypress.

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

- src/events/EventListener.h

4.11 Tedm::Logger Class Reference

Logger for debugging.

`#include <Logger.h>`

Public Types

- enum **LogLevel** {
  LOG_DEBUG = 1, **LOG_INFO** = 2, **LOG_WARN** = 3, **LOG_ERROR** = 4,
  LOG_NONE = 5 }

Public Member Functions

- Logger ()
  default constructor

Static Public Member Functions

- static void log (LogLevel level, std::string msg)
  log a message
- static void log_error (std::string msg)
  log a message at error level
- static void log_warning (std::string msg)
  log a message at warning level
- static void log_info (std::string msg)
  log a message at info level
- static void log_debug (std::string msg)
  log a message at debug level
- static LogLevel getLevel ()
  Retrieve the current logging level.
- static void setLevel (LogLevel level)
  Set the current logging level.

Static Public Attributes

- static const std::string **level_strings** []

4.11.1 Detailed Description

Logger for debugging.

4.11.2 Member Function Documentation

4.11.2.1 Logger::LogLevel Tedm::Logger::getLevel ( ) [static]

Retrieve the current logging level.

Returns

the current logging level

4.11.2.2 void Tedm::Logger::log (Logger::LogLevel *msg_level,* std::string *msg*) [static]

log a message

Parameters

level	the logging level
msg	the message

4.11.2.3 void Tedm::Logger::log_debug (std::string *msg*) [static]

log a message at debug level

Parameters

msg	the message

4.11.2.4 void Tedm::Logger::log_error (std::string *msg*) [static]

log a message at error level

Parameters

msg	the message

4.11.2.5 void Tedm::Logger::log_info (std::string *msg*) [static]

log a message at info level

Parameters

msg	the message

4.11.2.6 void Tedm::Logger::log_warning (std::string *msg*) [static]

log a message at warning level

Parameters

msg	the message

4.11.2.7 void Tedm::Logger::setLevel (Logger::LogLevel *level*) [static]

Set the current logging level.

Parameters

level	the new logging level

4.11.3 Member Data Documentation

4.11.3.1 const std::string Tedm::Logger::level_strings [static]

Initial value:

```
= {
        "DEBUG",
        "INFO",
        "WARN",
        "ERROR",
        "NONE"
    }
```

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

- src/utils/Logger.h
- src/utils/Logger.cpp

4.12 Tedm::MouseButtonListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse button press.

`#include <EventListener.h>`

Public Member Functions

- virtual void **operator()** (int x, int y)=0

4.12.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse button press.

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

- src/events/EventListener.h

4.13 Tedm::MouseMoveListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse movement.

`#include <EventListener.h>`

Public Member Functions

- virtual void **operator()** (int x, int y, int rel_x, int rel_y, bool left_click, bool right_click, bool middle_click)=0

4.13.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse movement.

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

- src/events/EventListener.h

4.14 Tedm::MouseWheelListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse wheel interaction.

`#include <EventListener.h>`

Public Member Functions

- virtual void **operator()** (bool scroll_up, bool scroll_down)=0

4.14.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse wheel interaction.

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

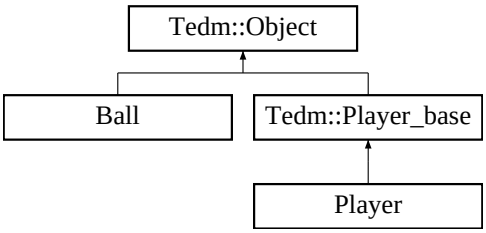
- src/events/EventListener.h

4.15 Tedm::Object Class Reference

Basic game element. Any item in game should inherit object.

`#include <object.h>`

Inheritance diagram for Tedm::Object:



Public Member Functions

- Object (Graphics &g)

  Constructor sets default position to (0,0), default image location in sprite to (0,0), and default size to (0,0)
- Object (Graphics &g, std::string filename)

  Constructor sets default position to (0,0), default image location in sprite to (0,0), and default size to (0,0) Includes filename to set image for sprite.
- Object (Graphics &g, const int x, const int y, const int h, const int w)

Constructor with position and size.

- Object (Graphics &g, std::string filename, const int x, const int y, const int h, const int w)

  Constructor with position and size Includes filename to set image for sprite.
- void set_position (int x, int y)

  set position of the sprite on the screen
- bool collision (Object &obj)

  check if a collision has occured with another object
- int get_height ()

  Retrieve the height.
- int get_width ()

  Retrieve the width.
- int get_x ()

  Retrieve the x coordinate position.
- int get_y ()

  Retrieve the y coordinate position.
- void set_x (int i)

  Set the x coordinate position.
- void set_y (int i)

  Set the y coordinate position.
- void set_sprite (std::string filename)

  Set the sprite image.
- void draw ()

  render the sprite on the screen

Public Attributes

- Sprite_base **sprite**

Protected Attributes

- *the position of the object on the scree*

The size dimensions of the object.

4.15.1 Detailed Description

Basic game element. Any item in game should inherit object.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Tedm::Object::Object (Graphics & *g*) [inline]

Constructor sets default position to (0,0), default image location in sprite to (0,0), and default size to (0,0)

Parameters

g	the Graphics object for the game

4.15.2.2 Tedm::Object::Object (Graphics & *g*, std::string *filename*) [inline]

Constructor sets default position to (0,0), default image location in sprite to (0,0), and default size to (0,0) Includes filename to set image for sprite.

Parameters

g	the Graphics object for the game
filename	the file path for the image

4.15.2.3 Tedm::Object::Object (Graphics & *g*, const int *x*, const int *y*, const int *h*, const int *w*) [inline]

Constructor with position and size.

Parameters

g	the Graphics object for the game
x	the x coordinate screen location
y	the y coordinate screen location
h	the object height
w	the object width

4.15.2.4 Tedm::Object::Object (Graphics & *g*, std::string *filename*, const int *x*, const int *y*, const int *h*, const int *w*) [inline]

Constructor with position and size Includes filename to set image for sprite.

Parameters

g	the Graphics object for the game
x	the x coordinate screen location
y	the y coordinate screen location
h	the object height
w	the object width
filename	the file path for the image

4.15.3 Member Function Documentation

4.15.3.1 bool Tedm::Object::collision (Object & *obj*) [inline]

check if a collision has occured with another object

Parameters

obj	the object to check for collision

Returns

true if collision has occurred

4.15.3.2 int Tedm::Object::get_height ( ) [inline]

Retrieve the height.

Returns

the object height

4.15.3.3 int Tedm::Object::get_width ( ) [inline]

Retrieve the width.

Returns

the object width

4.15.3.4 int Tedm::Object::get_x ( ) [inline]

Retrieve the x coordinate position.

Returns

the x coordinate position

4.15.3.5 int Tedm::Object::get_y ( ) [inline]

Retrieve the y coordinate position.

Returns

the y coordinate position

4.15.3.6 void Tedm::Object::set_position (int *x*, int *y*) [inline]

set position of the sprite on the screen

Parameters

x	the x coordinate
y	the y coordinate

4.15.3.7 void Tedm::Object::set_sprite (std::string *filename*) [inline]

Set the sprite image.

Parameters

filename	the path to the new image

4.15.3.8 void Tedm::Object::set_x (int *i*) [inline]

Set the x coordinate position.

Parameters

i	the new x coordinate position

4.15.3.9 void Tedm::Object::set_y (int *i*) [inline]

Set the y coordinate position.

Parameters

i	the new y coordinate position

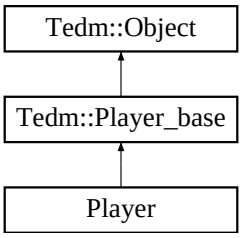
The documentation for this class was generated from the following file:

- src/objects/object.h

4.16 Player Class Reference

the user makes a Player class which inherits Player_base and defines functionality specific to their game. This includes default size, position, functions for user input, etc.

Inheritance diagram for Player:



Public Member Functions

- Player (Graphics &g, std::string filename, const int x, const int y)

  The Player inherits Player_base and sets the position.
- void set_pos (int x, int y)

  Sets the position of the player.
- void move_up ()

  Move up when the user enters the up key.
- void move_down ()

  Move down when the user enters the up key.
- int get_y ()

  Get the players y coordinate position.
- int get_height ()

  Get the height of the player object.

Additional Inherited Members

4.16.1 Detailed Description

the user makes a Player class which inherits Player_base and defines functionality specific to their game. This includes default size, position, functions for user input, etc.

See Also

Player_base

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Player::Player (Graphics & *g*, std::string *filename*, const int *x*, const int *y*) [inline]

The Player inherits Player_base and sets the position.

Parameters

x	The starting x coordinate
y	The starting y coordinate

4.16.3 Member Function Documentation

4.16.3.1 int Player::get_height ( ) [inline]

Get the height of the player object.

Returns

the height

4.16.3.2 int Player::get_y ( ) [inline]

Get the players y coordinate position.

Returns

the y coordinate position

4.16.3.3 void Player::set_pos (int *x*, int *y*) [inline]

Sets the position of the player.

Parameters

x	The starting x coordinate
y	The starting y coordinate

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

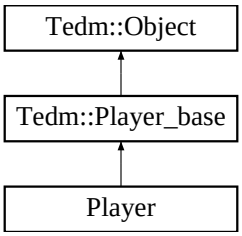
- demos/pong.cpp

4.17 Tedm::Player_base Class Reference

Define player object for game interaction The developer can inherit player to create custom players for each game.

`#include <player.h>`

Inheritance diagram for Tedm::Player_base:



Public Member Functions

- Player_base (Graphics &g, const int x, const int y, const int h, const int w)

  Constructor creates object to represent player.
- Player_base (Graphics &g, std::string filename, const int x, const int y, const int h, const int w)

  Constructor creates object to represent player Includes default image.

Additional Inherited Members

4.17.1 Detailed Description

Define player object for game interaction The developer can inherit player to create custom players for each game.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Tedm::Player_base::Player_base (Graphics & *g,* const int *x,* const int *y,* const int *h,* const int *w*) [inline]

Constructor creates object to represent player.

See Also

Object

4.17.2.2 Tedm::Player_base::Player_base (Graphics & *g,* std::string *filename,* const int *x,* const int *y,* const int *h,* const int *w*) [inline]

Constructor creates object to represent player Includes default image.

See Also

Object

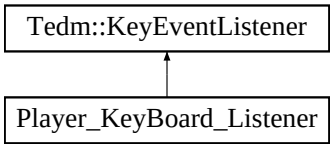
The documentation for this class was generated from the following file:

- src/objects/player.h

4.18 Player_KeyBoard_Listener Class Reference

The user greates a KeyEventListener class to define the functions that will execute in response to user input, inheriting the KeyEventListener class and overriding the appropriate virtual functions.

Inheritance diagram for Player_KeyBoard_Listener:



Public Member Functions

- **Player_KeyBoard_Listener** (Player &p1, Player &p2)
- void **operator()** (SDL_Keycode sym) override

4.18.1 Detailed Description

The user greates a KeyEventListener class to define the functions that will execute in response to user input, inheriting the KeyEventListener class and overriding the appropriate virtual functions.

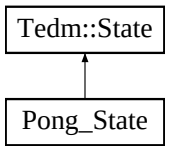
The documentation for this class was generated from the following file:

- demos/pong.cpp

4.19 Pong_State Class Reference

The user creates State class for each game state which inherits the State class and implements elements specific to the state.

Inheritance diagram for Pong_State:



Public Member Functions

- Pong_State (Game &game)

  The constructor initializes the paddles and ball.
- void new_round ()

  reset positions of paddle and ball for new round
- bool init () override

  Override the default init() function to set default conditions for a new game.
- void destroy () override

  Remove state event listeners so they do not interfere with the following state.
- void paused () override

  Pause the game.
- void resumed () override

  Unpause the game.
- void update () override

  the game will call the update() function every frame, executing the main functionality of the game. Primary game logic goes here
- void render () override

  the game will call render after update each frame. This function draws everything relevant for the current state

Public Attributes

- Player **p1**
- Player **p2**
- Ball **ball**
- SDL_Texture * **background**

Additional Inherited Members

4.19.1 Detailed Description

The user creates State class for each game state which inherits the State class and implements elements specific to the state.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Pong_State::Pong_State (Game & *game*) [inline]

The constructor initializes the paddles and ball.

Parameters

game	the main Game object

4.19.3 Member Function Documentation

4.19.3.1 bool Pong_State::init ( ) [inline], [override], [virtual]

Override the default init() function to set default conditions for a new game.

Returns

true if success

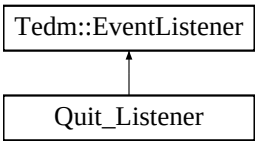
Implements Tedm::State.

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

- demos/pong.cpp

4.20 Quit_Listener Class Reference

Inheritance diagram for Quit_Listener:



Public Member Functions

- **Quit_Listener** (bool &b)
- void **operator()** () override

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

- demos/pong.cpp

4.21 Tedm::Sprite_base Class Reference

Graphic representation of game element.

`#include <sprite.h>`

Public Member Functions

- Sprite_base (Graphics &g)

  Constructor sets size to 0 and contains no image.
- Sprite_base (Graphics &g, int height, int width)

  Constructor sets size without image.
- Sprite_base (Graphics &g, std::string filename, int height, int width)

  Constructor sets size and image.
- ~Sprite_base ()

  Destructor cleans up the texture.
- void set_sprite (std::string filename)

  Sets the sprite image.
- void set_height_width (int height, int width)

  Set the size of the sprite.
- SDL_Texture * get_sprite ()

  Retrieve the underlying sprite object.
- void set_source_pos (int x, int y)

  set the location of the sprite within the image
- void set_position (int x, int y)

  set the position of the sprite on the screen
- SDL_Rect * get_pos ()

  Get the sprite location on the screen.
- SDL_Rect * get_src ()

  Get the sprite location in the image.
- void draw ()

  Render the sprite on the screen.

Public Attributes

- SDL_Texture * **sprite**
- SDL_Rect **src**
- SDL_Rect **tgt**
- std::string **filename**
- Graphics & **graphics**

4.21.1 Detailed Description

Graphic representation of game element.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Tedm::Sprite_base::Sprite_base (Graphics & *g*) [inline]

Constructor sets size to 0 and contains no image.

Parameters

g	the game Graphics object

4.21.2.2 Tedm::Sprite_base::Sprite_base (Graphics & *g*, int *height*, int *width*) [inline]

Constructor sets size without image.

Parameters

height	the object height
width	the object width
g	the game Graphics object

4.21.2.3 Tedm::Sprite_base::Sprite_base (Graphics & *g*, std::string *filename*, int *height*, int *width*) [inline]

Constructor sets size and image.

Parameters

filename	the image to load
height	the object height
width	the object width
g	the game Graphics object

4.21.3 Member Function Documentation

4.21.3.1 SDL_Rect* Tedm::Sprite_base::get_pos ( ) [inline]

Get the sprite location on the screen.

Returns

the SDL_Rect object which contains location data

4.21.3.2 SDL_Texture* Tedm::Sprite_base::get_sprite ( ) [inline]

Retrieve the underlying sprite object.

Returns

the sprite texture

4.21.3.3 SDL_Rect* Tedm::Sprite_base::get_src ( ) [inline]

Get the sprite location in the image.

Returns

the SDL_Rect object which contains location data

4.21.3.4 void Tedm::Sprite_base::set_height_width (int *height*, int *width*) [inline]

Set the size of the sprite.

Parameters

height	the height
width	the width

4.21.3.5 void Tedm::Sprite_base::set_position (int *x*, int *y*) [inline]

set the position of the sprite on the screen

Parameters

x	the x coordinate location of the sprite on the screen
y	the y coordinate location of the sprite on the screen

4.21.3.6 void Tedm::Sprite_base::set_source_pos (int *x*, int *y*) [inline]

set the location of the sprite within the image

Parameters

x	the x coordinate of the sprite in the image
y	the y coordinate of the sprite in the image

4.21.3.7 void Tedm::Sprite_base::set_sprite (std::string *filename*) [inline]

Sets the sprite image.

Parameters

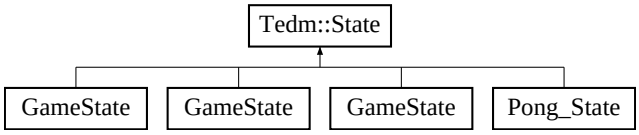
filename	the image file path

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

- src/objects/sprite.h

4.22 Tedm::State Class Reference

Inheritance diagram for Tedm::State:



Public Member Functions

- State (Game &game, std::string id)

  Constructor.
- virtual ~State ()

  Default destructor.
- std::string getID ()

  Retrieve the string identifier.
- virtual bool init ()=0

  virtual function to be overridden by game-specific state Contains logic executed at game start
- virtual void update ()=0

  virtual function to be overridden by game-specific state Contains logic executed at each frame
- virtual void render ()=0

  virtual function to be overridden by game-specific state Draws the state-appropriate items to the screen
- virtual void destroy ()=0

  virtual function to be overridden by game-specific state Cleans state listeners in preperation for new state
- virtual void paused ()=0

  virtual function to be overridden by game-specific state Pauses the game
- virtual void resumed ()=0

  virtual function to be overridden by game-specific state Un-pauses the game
- bool operator== (const State &other)

  Copy assignment.

Protected Attributes

- Game & **game**
- Graphics & **graphics**
- EventHandler & **eventHandler**
- Context & **context**
- std::string **id**

4.22.1 Constructor & Destructor Documentation

4.22.1.1 Tedm::State::State (Game & *game*, std::string *id*)

Constructor.

Parameters

game	the game parent
id	the string identifier of the state

4.22.2 Member Function Documentation

4.22.2.1 std::string Tedm::State::getID ( ) [inline]

Retrieve the string identifier.

Returns

the string

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

- src/State.h
- src/State.cpp

4.23 Tedm::Texture Class Reference

Define Texture to store object or background image.

`#include <Texture.h>`

4.23.1 Detailed Description

Define Texture to store object or background image.

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

- src/img/Texture.h

4.24 Timer Class Reference

Timer for maintaining frame rate.

`#include <Timer.h>`

Public Member Functions

- Timer ()
  Constructori.
- void start ()
  start the timer
- void stop ()
  stop the timer
- void pause ()
  pause the timer
- void unpause ()
  unpause the timer
- int get_ticks ()
  check the number of ticks since timer started
- bool is_started ()
  Checks the status of the timer.
- bool is_paused ()
  Check if timer is paused.

4.24.1 Detailed Description

Timer for maintaining frame rate.

4.24.2 Member Function Documentation

4.24.2.1 int Timer::get_ticks ( )

check the number of ticks since timer started

Returns

the quantity of ticks since timer started

4.24.2.2 bool Timer::is_paused ( )

Check if timer is paused.

Returns

true if timer is paused

4.24.2.3 bool Timer::is_started ( )

Checks the status of the timer.

Returns

true if timer is running

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

- src/utils/Timer.h
- src/utils/Timer.cpp

4.25 Tedm::UserListener Class Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event.

`#include <EventListener.h>`

Public Member Functions

- virtual void **operator()** (int type, int code, void ∗data1, void ∗data2)=0

4.25.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event.

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

- src/events/EventListener.h

Chapter 5

File Documentation

5.1 demos/pong.cpp File Reference

Simple example game demonstrating Tedm API.

```
#include <vector>
#include <math.h>
#include "Game.h"
#include "objects/player.h"
```

Classes

- class Player

  the user makes a Player class which inherits Player_base and defines functionality specific to their game. This includes default size, position, functions for user input, etc.
- class Ball

  the user makes a class for any object type in the game. In this case the ball is the only non-player object
- class Player_KeyBoard_Listener

  The user greates a KeyEventListener class to define the functions that will execute in response to user input, inheriting the KeyEventListener class and overriding the appropriate virtual functions.
- class Quit_Listener
- class Pong_State

  The user creates State class for each game state which inherits the State class and implements elements specific to the state.

Macros

- #define **M_PI** 3.14159265358979323846

Functions

- int **main** (int argc, char ∗argv[])

5.1.1 Detailed Description

Simple example game demonstrating Tedm API.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017 This is a basic implementation of Pong.

5.2 src/Context.cpp File Reference

Contains details about the game condition.

```
#include "Context.h"
```

5.2.1 Detailed Description

Contains details about the game condition.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

5.3 src/Context.h File Reference

Contains details about the game condition.

```
#include <string>
```

Classes

- class Tedm::Context

  Contains details about the game condition.

5.3.1 Detailed Description

Contains details about the game condition.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

5.4 src/events/Event.cpp File Reference

Create hooks for key presses and other in-game events that trigger changes in game state.

```
#include "Event.h"
```

5.4.1 Detailed Description

Create hooks for key presses and other in-game events that trigger changes in game state.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.5 src/events/Event.h File Reference

Create hooks for key presses and other in-game events that trigger changes in game state.

```
#include <SDL2/SDL_events.h>
```

Classes

- class Tedm::Event

  Create hooks for key presses and other in-game events that trigger changes in game state.

5.5.1 Detailed Description

Create hooks for key presses and other in-game events that trigger changes in game state.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.6 src/events/EventHandler.cpp File Reference

Create handlers for Events so developers can indicate which events their game will respond to.

```
#include "EventHandler.h"
```

5.6.1 Detailed Description

Create handlers for Events so developers can indicate which events their game will respond to.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

See Also

http://www.sdltutorials.com/sdl-events

5.7 src/events/EventHandler.h File Reference

Create handlers for Events so developers can indicate which events their game will respond to.

```
#include <vector>
#include <algorithm>
#include <memory>
#include "Event.h"
#include "EventListener.h"
#include "EventTrigger.h"
```

Classes

- class Tedm::EventHandler

5.7.1 Detailed Description

Create handlers for Events so developers can indicate which events their game will respond to.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

See Also

http://www.sdltutorials.com/sdl-events

5.8 src/events/EventListener.h File Reference

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event.

```
#include <SDL2/SDL_keycode.h>
```

Classes

- class Tedm::EventListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is the default case for custom events that are not SDL key events.
- class Tedm::KeyEventListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon keypress.
- class Tedm::MouseMoveListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse movement.
- class Tedm::MouseWheelListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse wheel interaction.
- class Tedm::MouseButtonListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event This is executed upon mouse button press.
- class Tedm::UserListener

  Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event.

5.8.1 Detailed Description

Define functions to occur upon event occurance. User can overload () operator and it will be executed with the event.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.9 src/Game.cpp File Reference

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

```
#include "Game.h"
```

5.9.1 Detailed Description

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

5.10 src/Game.h File Reference

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

```
#include <unordered_map>
#include "events/EventHandler.h"
#include "utils/Logger.h"
#include "objects/object.h"
#include "Context.h"
#include "State.h"
#include "Graphics.h"
#include "utils/Timer.h"
```

Classes

- class Tedm::Game

  Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

5.10.1 Detailed Description

Primary object in library. The game contains all other members which together represent a game. Developers can inherit this class to define a game.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

5.11 src/Graphics.cpp File Reference

The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects.

```
#include "Graphics.h"
```

5.11.1 Detailed Description

The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

See Also

www.lazyfoo.net/tutorials/SDL/index.php

5.12 src/Graphics.h File Reference

The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects.

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <iostream>
#include <string>
#include "utils/Logger.h"
```

Classes

- class Tedm::Graphics

5.12.1 Detailed Description

The user should be able to choose a graphics solution, so this class abstracts graphics away from the game library. The graphics object is passed to objects that will require rendering. It currently represents SDL and contains window and renderer objects.

Author

David Watkins, Theodore Ahlfeld, and Matthew Haigh

Date

27 April 2017

See Also

www.lazyfoo.net/tutorials/SDL/index.php

5.13 src/img/Texture.h File Reference

Define Texture to store object or background image.

```
#include <SDL_system.h>
```

Classes

- class Tedm::Texture

  Define Texture to store object or background image.

5.13.1 Detailed Description

Define Texture to store object or background image.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.14 src/objects/player.h File Reference

Basic game element. Any item in game should inherit object.

```
#include <string>
#include "objects/object.h"
```

Classes

- class Tedm::Player_base

  Define player object for game interaction The developer can inherit player to create custom players for each game.

5.14.1 Detailed Description

Basic game element. Any item in game should inherit object. Define player object for game interaction The developer can inherit player to create custom players for each game.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.15 src/objects/sprite.h File Reference

Graphic representation of game element.

```
#include <string>
#include <SDL2/SDL.h>
#include <fstream>
#include "Graphics.h"
```

Classes

- class Tedm::Sprite_base

  Graphic representation of game element.

5.15.1 Detailed Description

Graphic representation of game element.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.16 src/State.h File Reference

State object used for defining a game state.

```
#include <SDL2/SDL.h>
#include <string>
#include <events/EventHandler.h>
#include "Context.h"
#include "events/Event.h"
#include "Game.h"
```

Classes

- class Tedm::State

5.16.1 Detailed Description

State object used for defining a game state.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

5.16.2 DESCRIPTION

This c++ class defines the basis for a given game. Each game should define a state through which it enters and subsequently exits from.

5.17 src/utils/Timer.cpp File Reference

Timer for maintaining frame rate.

```
#include "utils/Timer.h"
```

5.17.1 Detailed Description

Timer for maintaining frame rate.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

See Also

http://lazyfoo.net/tutorials/SDL/14_animated_sprites_and_vsync/index.php

5.18 src/utils/Timer.h File Reference

Timer for maintaining frame rate.

```
#include <SDL2/SDL.h>
```

Classes

- class Timer

  Timer for maintaining frame rate.

5.18.1 Detailed Description

Timer for maintaining frame rate.

Author

David Watkins, Theodore Ahlfeld, Matthew Haigh

Date

4/27/2017

Version

1.0

See Also

http://lazyfoo.net/tutorials/SDL/14_animated_sprites_and_vsync/index.php