PROItheGame

Piotr Satała

Version

5/27/2019 11:32:00 PM

Table of Contents

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

Camera Class Reference

`#include <Camera.h>`

Public Member Functions

- **Camera** (int x=0, int y=0)
- **~Camera** ()
  destructor
- void **setXCoordinate** (int newValue)
  setter for x
- void **setYCoordinate** (int newValue)
  setter for y
- int **getXCoordinate** ()
  getter for x
- int **getYCoordinate** ()
  getter for y

Detailed Description

Name: **Camera.h** Purpose: declaration and definition of camera class Author: Piotr Satala

Definition at line 7 of file Camera.h.

Constructor & Destructor Documentation

Camera::Camera (int *x* = 0, int *y* = 0) [inline]

constructor parameters are: x and y coordinates of camera

Definition at line 18 of file Camera.h.

Camera::~Camera () [inline]

destructor

Definition at line 22 of file Camera.h.

Member Function Documentation

int Camera::getXCoordinate () [inline]

getter for x

Definition at line 40 of file Camera.h.

int Camera::getYCoordinate () [inline]

getter for y

Definition at line 43 of file Camera.h.

void Camera::setXCoordinate (int *newValue*) [inline]

setter for x

Definition at line 28 of file Camera.h.

void Camera::setYCoordinate (int *newValue*) [inline]

setter for y

Definition at line 31 of file Camera.h.

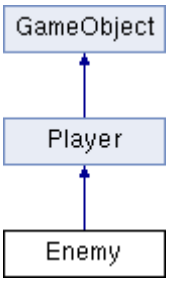
The documentation for this class was generated from the following file:

- PROItheGame/**Camera.h**

Enemy Class Reference

`#include <Enemy.h>`

Inheritance diagram for Enemy:



Public Member Functions

- **Enemy** ()
  default constructor
- **Enemy** (int x, int y, int w, int h, double xC, double yC, double g, int tDist, int tTime, std::string behaviourType="")
  parametrised constructor
- virtual **~Enemy** ()
  destructor
- virtual void **setColor** ()
  function responsible for setting the color to print in
- void **checkCollision** (**HumanPlayer** *myPlayer)
- void **applyBehaviour** (std::vector< **GameObject** *> myVector)
  function responsible for handling the enemy behaviour
- void **checkCollision** (**Obstacle** *obstacle)
  *using function from class **Player** to check collision with obstacles*

Additional Inherited Members

Detailed Description

Name: **Enemy.h** Purpose: declaration of **Enemy** class Author: Piotr Satala

Definition at line 12 of file Enemy.h.

Constructor & Destructor Documentation

Enemy::Enemy () `[inline]`

default constructor

Definition at line 60 of file Enemy.h.

Enemy::Enemy (int *x*, int *y*, int *w*, int *h*, double *xC*, double *yC*, double *g*, int *tDist*, int *tTime*, std::string *behaviourType* = "") `[inline]`

parametrised constructor

Definition at line 64 of file Enemy.h.

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

destructor

Definition at line 96 of file Enemy.h.

Member Function Documentation

void Enemy::applyBehaviour (std::vector< GameObject *> *myVector*) `[inline], [virtual]`

function responsible for handling the enemy behaviour

Reimplemented from **Player** (*p.28*).

Definition at line 115 of file Enemy.h.

void Enemy::checkCollision (HumanPlayer * *myPlayer*)

function responsible for checking if enemy collided with a player and reacting accordingly to whatever part hit the player parameters are: player to check

Definition at line 83 of file Enemy.cpp.

void Player::checkCollision

using function from class **Player** to check collision with obstacles

Definition at line 114 of file Player.cpp.

void Enemy::setColor () `[virtual]`

function responsible for setting the color to print in

Reimplemented from **Player** (*p.30*).

Definition at line 75 of file Enemy.cpp.

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

- PROItheGame/**Enemy.h**
- PROItheGame/**Enemy.cpp**

Game Class Reference

`#include <Game.h>`

Public Member Functions

- **Game** ()
  default constructor
- **Game** (int screenHeight, int screenWidth, int menuHeight, int menuWidth, double timeBetweenFrames)
- **~Game** ()
  destructor
- void **simulateMenu** ()
  function responsbile for simulating interaction with the menu
- void **handleMenuChoice** (int functionID)
- void **clear** ()
  function responsible for clearing the screen
- void **close** ()
  function responsible for closing the game

Detailed Description

Name: **Game.h** Purpose: declaration of **Game** class, which is responsible for handling the entire application Author: Piotr Satala

Definition at line 22 of file Game.h.

Constructor & Destructor Documentation

Game::Game () `[inline]`

default constructor

Definition at line 91 of file Game.h.

Game::Game (int *screenHeight*, int *screenWidth*, int *menuHeight*, int *menuWidth*, double *timeBetweenFrames*) `[inline]`

parametrised constructor parameters are: height and width of the screen, height and width of each menu element, time between two frames

Definition at line 96 of file Game.h.

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

destructor

Definition at line 109 of file Game.h.

Member Function Documentation

void Game::clear ()

function responsible for clearing the screen

Definition at line 635 of file Game.cpp.

void Game::close ()

function responsible for closing the game

Definition at line 645 of file Game.cpp.

void Game::handleMenuChoice (int *functionID*)

function responsible for responding to user's interaction with the menu parameters are: ID of function returned by element of menu chosen by the user

Definition at line 208 of file Game.cpp.

void Game::simulateMenu ()

function responsbile for simulating interaction with the menu

Definition at line 143 of file Game.cpp.

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

- PROItheGame/**Game.h**
- PROItheGame/**Game.cpp**

GameArea Class Reference

`#include <GameArea.h>`

Public Member Functions

- **GameArea** (int x=0, int y=0, int w=0, int h=0)
- **~GameArea** ()
  destructor
- void **checkIfInside** (**HumanPlayer** *myPlayer)

Static Public Member Functions

- static int **getCount** ()
  getter for count

Detailed Description

Name: **GameArea.h** Purpose: declaration of **GameArea** class Author: Piotr Satala

Definition at line 11 of file GameArea.h.

Constructor & Destructor Documentation

GameArea::GameArea (int *x* = 0, int *y* = 0, int *w* = 0, int *h* = 0)

constructor parameters are: x and y coordinates of the upper left corner, width and height of the rectangle indicating game area

Definition at line 14 of file GameArea.cpp.

GameArea::~GameArea () [inline]

destructor

Definition at line 25 of file GameArea.h.

Member Function Documentation

void GameArea::checkIfInside (HumanPlayer * *myPlayer*)

function responsible for checking if the user's character has left game area parameters are: pointer to an instance of human player - object controlled by the user function throws an exception if user is outside the game area

Definition at line 25 of file GameArea.cpp.

static int GameArea::getCount () [inline], [static]

getter for count

Definition at line 36 of file GameArea.h.

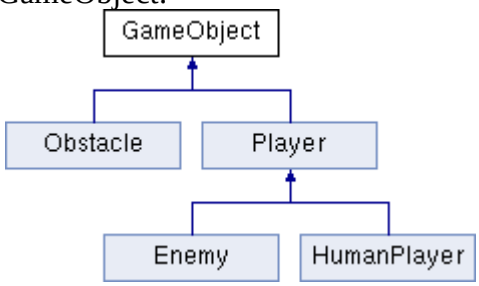
The documentation for this class was generated from the following files:

- PROItheGame/**GameArea.h**
- PROItheGame/**GameArea.cpp**

GameObject Class Reference

`#include <GameObject.h>`

Inheritance diagram for GameObject:



Public Types

- enum **Direction** { **UP**, **DOWN**, **LEFT**, **RIGHT** }

enum indicating direction Public Member Functions

- virtual void **OverrideMe** ()=0
  function ensures the class is abstract
- **GameObject** (int x=0, int y=0, int w=0, int h=0)
- virtual **~GameObject** ()
  destructor
- void **setXCoordinate** (double newValue)
  setter for x coordinate
- void **setYCoordinate** (double newValue)
  setter for y coordinate
- void **setObjectHeight** (int newValue)
  setter for object's height
- void **setObjectWidth** (int newValue)
  setter for object's width
- void **setIsAlive** (bool newValue)
  setter for isAlive
- double **getXCoordinate** ()
  getter for x coordinate
- double **getYCoordinate** ()
  getter for y coordinate
- int **getObjectHeight** ()
  gettter for object's height
- int **getObjectWidth** ()
  getter for object's width
- bool **getIsAlive** ()
  getter for isAlive
- virtual void **calculateNextPosition** (const double timeDifference)
- virtual void **setColor** ()
- virtual void **applyBehaviour** (std::vector< **GameObject** *> myVector)
- bool **checkCollisionSide** (**GameObject** *obstacle, **Direction** dir)
- void **print** (SDL_Renderer *renderer, **Camera** *myCamera)

Protected Attributes

- double **xCoordinate**

- double **yCoordinate**
- int **objectHeight**
- int **objectWidth**
- std::tuple< unsigned int, unsigned int, unsigned int > **color**
- bool **isAlive** = true

Detailed Description

Name: **GameObject.h** Purpose: declaration and definition of abstract **GameObject** class Author: Piotr Satala

Definition at line 17 of file GameObject.h.

Member Enumeration Documentation

enum GameObject::Direction

enum indicating direction

Enumerator:

UP	
DOWN	
LEFT	
RIGHT	

Definition at line 44 of file GameObject.h.

Constructor & Destructor Documentation

GameObject::GameObject (int *x* = 0, int *y* = 0, int *w* = 0, int *h* = 0) [inline]

constructor parameters are: x and y coordinates, width and height of the object

Definition at line 50 of file GameObject.h.

virtual GameObject::~GameObject () [inline], [virtual]

destructor

Definition at line 55 of file GameObject.h.

Member Function Documentation

virtual void GameObject::applyBehaviour (std::vector< GameObject *> *myVector*) [inline], [virtual]

function responsible for applying behaviour for the player parameters are: vector of all objects function body is empty, since it is made to be overridden

Reimplemented in **Enemy** (*p.8*), **Player** (*p.28*), and **HumanPlayer** (*p.19*).

Definition at line 116 of file GameObject.h.

virtual void GameObject::calculateNextPosition (const double *timeDifference*) [inline], [virtual]

function responsible for calculating next position of the object after given time parameters are: time which passed since previous position function body is empty, since it is made to be overridden

Reimplemented in **Player** (*p.28*).

Definition at line 105 of file GameObject.h.

bool GameObject::checkCollisionSide (GameObject * *obstacle*, Direction *dir*)

function responsible for checking if player's side collided with an obstacle parameters are: obstacle to check, side to check

Definition at line 12 of file GameObject.cpp.

bool GameObject::getIsAlive () [inline]

getter for isAlive

Definition at line 95 of file GameObject.h.

int GameObject::getObjectHeight () [inline]

gettter for object's height

Definition at line 89 of file GameObject.h.

int GameObject::getObjectWidth () [inline]

getter for object's width

Definition at line 92 of file GameObject.h.

double GameObject::getXCoordinate () [inline]

getter for x coordinate

Definition at line 83 of file GameObject.h.

double GameObject::getYCoordinate () [inline]

getter for y coordinate

Definition at line 86 of file GameObject.h.

virtual void GameObject::OverrideMe () [pure virtual]

function ensures the class is abstract

Implemented in **Player** (*p.29*), and **Obstacle** (*p.26*).

void GameObject::print (SDL_Renderer * *renderer*, Camera * *myCamera*)

function responsible for printing the object onto the screen in relation to camera position parameters are: renderer to print on, camera to relate to

Definition at line 54 of file GameObject.cpp.

virtual void GameObject::setColor () `[inline], [virtual]`

function responsible for setting the color to print in function body is empty, since it is made to be overridden

Reimplemented in **Enemy** (*p.8*), **Player** (*p.30*), **Obstacle** (*p.26*), and **HumanPlayer** (*p.19*).

Definition at line 110 of file GameObject.h.

void GameObject::setIsAlive (bool *newValue*) `[inline]`

setter for isAlive

Definition at line 76 of file GameObject.h.

void GameObject::setObjectHeight (int *newValue*) `[inline]`

setter for object's height

Definition at line 70 of file GameObject.h.

void GameObject::setObjectWidth (int *newValue*) `[inline]`

setter for object's width

Definition at line 73 of file GameObject.h.

void GameObject::setXCoordinate (double *newValue*) `[inline]`

setter for x coordinate

Definition at line 64 of file GameObject.h.

void GameObject::setYCoordinate (double *newValue*) `[inline]`

setter for y coordinate

Definition at line 67 of file GameObject.h.

Member Data Documentation

std::tuple<unsigned int, unsigned int, unsigned int> GameObject::color `[protected]`

Definition at line 30 of file GameObject.h.

bool GameObject::isAlive = true `[protected]`

Definition at line 33 of file GameObject.h.

int GameObject::objectHeight `[protected]`

Definition at line 26 of file GameObject.h.

int GameObject::objectWidth `[protected]`

Definition at line 27 of file GameObject.h.

double GameObject::xCoordinate `[protected]`

Definition at line 22 of file GameObject.h.

double GameObject::yCoordinate `[protected]`

Definition at line 23 of file GameObject.h.

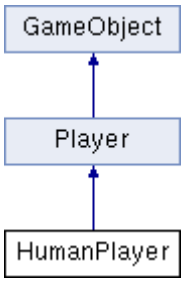
The documentation for this class was generated from the following files:

- PROItheGame/**GameObject.h**
- PROItheGame/**GameObject.cpp**

HumanPlayer Class Reference

`#include <HumanPlayer.h>`

Inheritance diagram for HumanPlayer:



Public Member Functions

- **HumanPlayer** ()
  default constructor
- **HumanPlayer** (int x, int y, int w, int h, double xC, double yC, double g, int tDist, int tTime)
- virtual **~HumanPlayer** ()
  destructor
- virtual void **setColor** ()
  function responsible for setting the color to print in
- virtual void **applyBehaviour** (std::vector< **GameObject** *> myVector)

Static Public Member Functions

- static int **getCount** ()
  getter for count

Additional Inherited Members

Detailed Description

Name: **HumanPlayer.h** Purpose: declaration of class **HumanPlayer** - its instance is controlled by the user Author: Piotr Satala

Definition at line 9 of file HumanPlayer.h.

Constructor & Destructor Documentation

HumanPlayer::HumanPlayer () `[inline]`

default constructor

Definition at line 19 of file HumanPlayer.h.

HumanPlayer::HumanPlayer (int *x*, int *y*, int *w*, int *h*, double *xC*, double *yC*, double *g*, int *tDist*, int *tTime*) `[inline]`

parametrised constructor parameters are object's: x coordinate, y coordinate, height, width, constant of speed in x and in y, gravity constant, distance covered by teleporting once and minimal time between teleports

Definition at line 24 of file HumanPlayer.h.

virtual HumanPlayer::~HumanPlayer () [inline], [virtual]

destructor

Definition at line 31 of file HumanPlayer.h.

Member Function Documentation

void HumanPlayer::applyBehaviour (std::vector< GameObject *> *myVector*) [virtual]

function responsible for handling keyboard inputs for the player parameters are: vector of all objects

Reimplemented from **Player** (*p.28*).

Definition at line 20 of file HumanPlayer.cpp.

static int HumanPlayer::getCount () [inline], [static]

getter for count

Definition at line 49 of file HumanPlayer.h.

void HumanPlayer::setColor () [virtual]

function responsible for setting the color to print in

Reimplemented from **Player** (*p.30*).

Definition at line 14 of file HumanPlayer.cpp.

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

- PROItheGame/**HumanPlayer.h**
- PROItheGame/**HumanPlayer.cpp**

MenuObject Class Reference

`#include <MenuObject.h>`

Public Member Functions

- void **OverrideMe** ()
  function overriding abstract function from base class
- **MenuObject** (int h=0, int w=0, std::string t="", int returnVal=-1)
- **~MenuObject** ()
  destructor
- void **print** (SDL_Renderer *rendererToPrintOn, int elementIndex, int elementCount)
- bool **checkIfClicked** (int xMouse, int yMouse)
- int **returnHere** ()

Detailed Description

Name: **MenuObject.h** Purpose: declaration of **MenuObject** class - used for creating menu Author: Piotr Satala

Definition at line 16 of file MenuObject.h.

Constructor & Destructor Documentation

MenuObject::MenuObject (int *h* = 0, int *w* = 0, std::string *t* = "", int *returnVal* = -1)

constructor parameters are: height, width, text of menu object and its return value

Name: **MenuObject.cpp** Purpose: definition of methods from **MenuObject** class Author: Piotr Satala

Definition at line 11 of file MenuObject.cpp.

MenuObject::~MenuObject ()

destructor

Definition at line 22 of file MenuObject.cpp.

Member Function Documentation

bool MenuObject::checkIfClicked (int *xMouse*, int *yMouse*)

function responsible for checking if user clicked the object parameters are: x and y coordinates of a click

Definition at line 72 of file MenuObject.cpp.

void MenuObject::OverrideMe () [inline]

function overriding abstract function from base class

Definition at line 29 of file MenuObject.h.

void MenuObject::print (SDL_Renderer * *rendererToPrintOn*, int *elementIndex*, int *elementCount*)

function responsible for printing the object of menu onto the screen in a correct place parameters are: renderer to print on, index of element from a vector, total count of elements in that part of menu

Definition at line 28 of file MenuObject.cpp.

int MenuObject::returnHere ()

function responsible for getting the return value of element function returns the return value of the element

Definition at line 80 of file MenuObject.cpp.

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

- PROItheGame/**MenuObject.h**
- PROItheGame/**MenuObject.cpp**

Momentum< T1, T2, T3 > Class Template Reference

`#include <Momentum.h>`

Public Member Functions

- **Momentum** (T3 g=0, T1 Vx=0, T2 Vy=0)
  constructor
- **Momentum** (const **Momentum** &other)
  copy constructor
- **~Momentum** ()
  destructor
- void **setXVelocity** (T1 newValue)
  setters
- void **setYVelocity** (T2 newValue)
- void **setGForce** (T3 newValue)
- T1 **getXVelocity** ()
  getters
- T2 **getYVelocity** ()
- T3 **getGForce** ()
- **Momentum** & **operator=** (const **Momentum** &other)
  assingment operator

Detailed Description

template<typename T1, typename T2, typename T3>

class Momentum< T1, T2, T3 >

Name: **Momentum.h** Purpose: declaration of functions covering momentum of a given object in x and y directions Author: Piotr Satala

Definition at line 8 of file Momentum.h.

Constructor & Destructor Documentation

template<typename T1, typename T2, typename T3> Momentum< T1, T2, T3 >::Momentum (T3 *g* = 0, T1 *Vx* = 0, T2 *Vy* = 0) [inline]

constructor

Definition at line 18 of file Momentum.h.

template<typename T1, typename T2, typename T3> Momentum< T1, T2, T3 >::Momentum (const Momentum< T1, T2, T3 > & *other*) [inline]

copy constructor

Definition at line 22 of file Momentum.h.

template<typename T1, typename T2, typename T3> Momentum< T1, T2, T3 >::~Momentum () `[inline]`

destructor

Definition at line 29 of file Momentum.h.

Member Function Documentation

template<typename T1, typename T2, typename T3> T3 Momentum< T1, T2, T3 >::getGForce () `[inline]`

Definition at line 45 of file Momentum.h.

template<typename T1, typename T2, typename T3> T1 Momentum< T1, T2, T3 >::getXVelocity () `[inline]`

getters

Definition at line 43 of file Momentum.h.

template<typename T1, typename T2, typename T3> T2 Momentum< T1, T2, T3 >::getYVelocity () `[inline]`

Definition at line 44 of file Momentum.h.

template<typename T1, typename T2, typename T3> Momentum& Momentum< T1, T2, T3 >::operator= (const Momentum< T1, T2, T3 > & *other*) `[inline]`

assingment operator

Definition at line 49 of file Momentum.h.

template<typename T1, typename T2, typename T3> void Momentum< T1, T2, T3 >::setGForce (T3 *newValue*) `[inline]`

Definition at line 37 of file Momentum.h.

template<typename T1, typename T2, typename T3> void Momentum< T1, T2, T3 >::setXVelocity (T1 *newValue*) `[inline]`

setters

Definition at line 35 of file Momentum.h.

template<typename T1, typename T2, typename T3> void Momentum< T1, T2, T3 >::setYVelocity (T2 *newValue*) `[inline]`

Definition at line 36 of file Momentum.h.

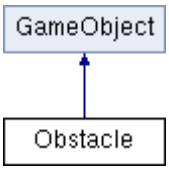
The documentation for this class was generated from the following file:

- PROItheGame/**Momentum.h**

Obstacle Class Reference

`#include <Obstacle.h>`

Inheritance diagram for Obstacle:



Public Member Functions

- void **OverrideMe** ()
  function overriding abstract function from base class
- **Obstacle** (int x=0, int y=0, int w=0, int h=0, bool canKill=false, bool isFinish=false)
- **~Obstacle** ()
  destructor
- virtual void **setColor** ()
  function responsible for setting the color to print in
- void **setCanItKill** (bool newValue)
  setter for can this object kill
- void **setIsItFinish** (bool newValue)
  setter for is it finish line
- bool **getCanItKill** ()
  getter for can this object kill
- bool **getIsItFinish** ()
  getter for is it finish line

Additional Inherited Members

Detailed Description

Name: **Obstacle.h** Purpose: declaration of obstacle class Author: Piotr Satala

Definition at line 11 of file Obstacle.h.

Constructor & Destructor Documentation

Obstacle::Obstacle (int *x* = `0`, int *y* = `0`, int *w* = `0`, int *h* = `0`, bool *canKill* = `false`, bool *isFinish* = `false`) `[inline]`

constructor parameters are: x and y coordinates, width and height, information about whether or no can it kill and is it a finish line

Definition at line 29 of file Obstacle.h.

Obstacle::~Obstacle () `[inline]`

destructor

Definition at line 41 of file Obstacle.h.

Member Function Documentation

bool Obstacle::getCanItKill () [inline]

getter for can this object kill

Definition at line 65 of file Obstacle.h.

bool Obstacle::getIsItFinish () [inline]

getter for is it finish line

Definition at line 68 of file Obstacle.h.

void Obstacle::OverrideMe () [inline], [virtual]

function overriding abstract function from base class

Implements **GameObject** (*p.15*).

Definition at line 24 of file Obstacle.h.

void Obstacle::setCanItKill (bool *newValue*) [inline]

setter for can this object kill

Definition at line 53 of file Obstacle.h.

void Obstacle::setColor () [virtual]

function responsible for setting the color to print in

Name: **Obstacle.cpp** Purpose: definition of methods from obstacle class Author: Piotr Satala

Reimplemented from **GameObject** (*p.16*).

Definition at line 9 of file Obstacle.cpp.

void Obstacle::setIsItFinish (bool *newValue*) [inline]

setter for is it finish line

Definition at line 56 of file Obstacle.h.

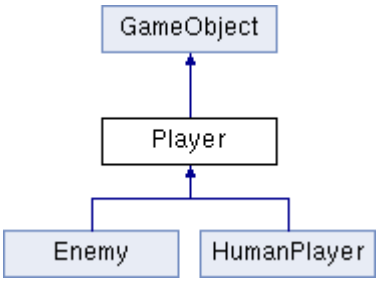
The documentation for this class was generated from the following files:

- PROItheGame/**Obstacle.h**
- PROItheGame/**Obstacle.cpp**

Player Class Reference

`#include <Player.h>`

Inheritance diagram for Player:



Public Member Functions

- void **OverrideMe** ()
  function overriding abstract function from base class
- **Player** ()
  default constructor
- **Player** (int x, int y, int w, int h, double xC, double yC, double g, int tDist, int tTime)
- virtual **~Player** ()
  destructor
- virtual void **setColor** ()
  function responsible for setting the color to print in
- virtual void **applyBehaviour** (std::vector< **GameObject** *> myVector)
- void **calculateNextPosition** (const double timeDifference)
- void **checkCollision** (**Obstacle** *obstacle)
- double **getXConstant** ()
  getter for x constant of velocity
- double **getYConstant** ()
  getter for y constant of velocity
- **Momentum**< double, double, double > * **getPlayerMomentum** ()
  getter for player's momentum
- bool **getHasFinished** ()
  getter for hasFinished
- void **setPlayerMomentum** (**Momentum**< double, double, double > newValue)
  setter for player momentum
- void **setHasFinished** (bool newValue)
  setter for hasFinished

Protected Member Functions

- void **jump** ()
  function responsible for jumping
- void **moveRight** ()
  function responsbile for moving to the left
- void **moveLeft** ()
  funstion responsbile for moving to the right
- void **stopX** ()
  function responsible for stopping the player in x axis
- void **teleport** (**Direction** dir, std::vector< **GameObject** *> myVector)

Protected Attributes

- bool **contact** [4] = { 0 }
  contact in each direction

Additional Inherited Members

Detailed Description

Definition at line 17 of file Player.h.

Constructor & Destructor Documentation

Player::Player () `[inline]`

default constructor

Definition at line 75 of file Player.h.

Player::Player (int *x*, int *y*, int *w*, int *h*, double *xC*, double *yC*, double *g*, int *tDist*, int *tTime*) `[inline]`

parametrised constructor parameters are object's: x coordinate, y coordinate, height, width, constant of speed in x and in y, gravity constant, distance covered by teleporting once and minimal time between teleports

Definition at line 79 of file Player.h.

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

destructor

Definition at line 95 of file Player.h.

Member Function Documentation

virtual void Player::applyBehaviour (std::vector< GameObject *> *myVector*) `[inline]`, `[virtual]`

function responsible for applying behaviour for the player parameters are: vector of all objects function body is empty, since its made to be overridden

Reimplemented from **GameObject** (*p.14*).

Reimplemented in **Enemy** (*p.8*), and **HumanPlayer** (*p.19*).

Definition at line 106 of file Player.h.

void Player::calculateNextPosition (const double *timeDifference*) `[virtual]`

function responsbile for calculating next position of the object based on previous position and speed parameters are: time between frames

Reimplemented from **GameObject** (*p.15*).

Definition at line 92 of file Player.cpp.

void Player::checkCollision (Obstacle * *obstacle*)

function responsible for checking if player collided with an obstacle and reacting accordingly to whatever part hit the obstacle parameters are: obstacle to check

Definition at line 114 of file Player.cpp.

bool Player::getHasFinished () `[inline]`

getter for hasFinished

Definition at line 136 of file Player.h.

Momentum<double, double, double>* Player::getPlayerMomentum () `[inline]`

getter for player's momentum

Definition at line 133 of file Player.h.

double Player::getXConstant () `[inline]`

getter for x constant of velocity

Definition at line 127 of file Player.h.

double Player::getYConstant () `[inline]`

getter for y constant of velocity

Definition at line 130 of file Player.h.

void Player::jump () `[protected]`

function responsible for jumping

Definition at line 35 of file Player.cpp.

void Player::moveLeft () `[protected]`

funstion responsbile for moving to the right

Definition at line 50 of file Player.cpp.

void Player::moveRight () `[protected]`

function responsbile for moving to the left

Definition at line 43 of file Player.cpp.

void Player::OverrideMe () `[inline]`, `[virtual]`

function overriding abstract function from base class

Implements **GameObject** (*p.15*).

Definition at line 71 of file Player.h.

virtual void Player::setColor () `[inline], [virtual]`

function responsible for setting the color to print in

Reimplemented from **GameObject** (*p.16*).

Reimplemented in **Enemy** (*p.8*), and **HumanPlayer** (*p.19*).

Definition at line 100 of file Player.h.

void Player::setHasFinished (bool *newValue*) `[inline]`

setter for hasFinished

Definition at line 147 of file Player.h.

void Player::setPlayerMomentum (Momentum< double, double, double > *newValue*) `[inline]`

setter for player momentum

Definition at line 144 of file Player.h.

void Player::stopX () `[protected]`

function responsible for stopping the player in x axis

Definition at line 56 of file Player.cpp.

void Player::teleport (Direction *dir*, std::vector< GameObject *> *myVector*) `[protected]`

function responsible for teleporting parameters are: direction in which teleport will take place, distance of the teleport, minimal time between two teleports in miliseconds and vector of obstacles

Definition at line 63 of file Player.cpp.

Member Data Documentation

bool Player::contact[4] = { 0 } `[protected]`

contact in each direction

Definition at line 48 of file Player.h.

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

- PROItheGame/**Player.h**
- PROItheGame/**Player.cpp**

Test Class Reference

`#include <Test.h>`

Public Member Functions

- **Test** ()
  constructor
- **~Test** ()
  destructor
- void **testAll** ()
  function calls all test methods

Detailed Description

Name: **Test.h** Purpose: declaration of **Test** class and all its methods Author: Piotr Satala

Definition at line 12 of file Test.h.

Constructor & Destructor Documentation

Test::Test () `[inline]`

constructor

Definition at line 51 of file Test.h.

Test::~Test () `[inline]`

destructor

Definition at line 54 of file Test.h.

Member Function Documentation

void Test::testAll ()

function calls all test methods

Definition at line 10 of file Test.cpp.

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

- UnitTestProject/**Test.h**
- UnitTestProject/**Test.cpp**

Tree< T > Class Template Reference

`#include <Tree.h>`

Public Member Functions

- **Tree** (T *firstElement=new T)
  constructor
- **Tree** (const **Tree**< T > &other)=delete
- **~Tree** ()
  destructor
- void **goTo** (int indexOfSon)
- void **goTo** (**TreeElement**< T > *newCurrentElement)
- void **add** (**TreeElement**< T > *addHere, T *newElement)
- void **add** (T *newElement)
- **Tree**< T > & **operator=** (const **Tree**< T > &other)=delete

Public Attributes

- **TreeElement**< T > * **ptrToCurrentElement**

Detailed Description

template<class T>

class Tree< T >

Name: **Tree.h** Purpose: declaration and definition of tree template Author: Piotr Satala

Definition at line 10 of file Tree.h.

Constructor & Destructor Documentation

template<class T> Tree< T >::Tree (T * *firstElement* = `new T`)

constructor

Definition at line 82 of file Tree.h.

template<class T> Tree< T >::Tree (const Tree< T > & *other*) `[delete]`

copy constructor constructing copies is not allowed

template<class T > Tree< T >::~Tree ()

destructor

Definition at line 92 of file Tree.h.

Member Function Documentation

template<class T> void Tree< T >::add (TreeElement< T > * *addHere*, T * *newElement*)

function responsible for adding new element to the given element of tree parameters are: pointer to element to which we will be adding, pointer to new object (to be added)

Definition at line 159 of file Tree.h.

template<class T> void Tree< T >::add (T * *newElement*)

function responsible for adding new element to current element parameters are: pointer to new object (to be added)

Definition at line 168 of file Tree.h.

template<class T > void Tree< T >::goTo (int *indexOfSon*)

function responsible for moving the current element to one of its sons or its father parameters are: index of son to go to or -1, if the target is father

Definition at line 132 of file Tree.h.

template<class T> void Tree< T >::goTo (TreeElement< T > * *newCurrentElement*)

function responsible for moving the current element to a new element of tree prameters are: pointer to new element of tree

Definition at line 147 of file Tree.h.

template<class T> Tree<T>& Tree< T >::operator= (const Tree< T > & *other*) `[delete]`

copy assignment operator making copies is not allowed

Member Data Documentation

template<class T> TreeElement<T>* Tree< T >::ptrToCurrentElement

Definition at line 21 of file Tree.h.

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

- PROItheGame/**Tree.h**

TreeElement< T > Class Template Reference

`#include <TreeElement.h>`

Public Member Functions

- **TreeElement** (T *objectToPointTo=NULL)
- **TreeElement** (const **TreeElement**< T > &other)=delete
- **~TreeElement** ()
  destructor
- **TreeElement**< T > & **operator=** (const **TreeElement**< T > &other)=delete

Public Attributes

- T * **ptrToObject**
- **TreeElement** * **ptrToFather**
- std::vector< **TreeElement** * > **listOfSons**

Detailed Description

template<class T>

class TreeElement< T >

Name: **TreeElement.h** Purpose: declaration and definition of element of tree template Author: Piotr Satala

Definition at line 11 of file TreeElement.h.

Constructor & Destructor Documentation

template<class T> TreeElement< T >::TreeElement (T * *objectToPointTo* = NULL)[inline]

constructor paramters are: pointer to an object element will be containing

Definition at line 28 of file TreeElement.h.

template<class T> TreeElement< T >::TreeElement (const TreeElement< T > & *other*)[delete]

copy constructor constructing copies is not allowed

template<class T > TreeElement< T >::~TreeElement ()

destructor

Definition at line 56 of file TreeElement.h.

Member Function Documentation

template<class T> TreeElement<T>& TreeElement< T >::operator= (const TreeElement< T > & *other*) `[delete]`

copy assignment operator making copies is not allowed

Member Data Documentation

template<class T> std::vector<TreeElement*> TreeElement< T >::listOfSons

Definition at line 20 of file TreeElement.h.

template<class T> TreeElement* TreeElement< T >::ptrToFather

Definition at line 18 of file TreeElement.h.

template<class T> T* TreeElement< T >::ptrToObject

Definition at line 16 of file TreeElement.h.

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

- PROItheGame/**TreeElement.h**

File Documentation

PROltheGame/Camera.h File Reference

Classes

- class **Camera**

PROItheGame/Enemy.cpp File Reference

```
#include "Enemy.h"
```

PROItheGame/Enemy.h File Reference

```
#include <string>
#include "HumanPlayer.h"
```

Classes

- class **Enemy**

PROItheGame/Game.cpp File Reference

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

PROItheGame/Game.h File Reference

```
#include <SDL.h>
#include <SDL_ttf.h>
#include <fstream>
#include <iostream>
#include <string>
#include "Enemy.h"
#include "HumanPlayer.h"
#include "Macros.h"
#include "Tree.h"
#include "MenuObject.h"
#include "GameArea.h"
```

Classes

- class **Game**

PROItheGame/GameArea.cpp File Reference

```
#include "GameArea.h"
```

PROItheGame/GameArea.h File Reference

```
#include <SDL.h>
#include "HumanPlayer.h"
```

Classes

- class **GameArea**

PROItheGame/GameObject.cpp File Reference

```
#include "GameObject.h"
```

PROItheGame/GameObject.h File Reference

```
#include <SDL.h>
#include <vector>
#include <tuple>
#include <math.h>
#include "Camera.h"
```

Classes

- class **GameObject**

PROItheGame/HumanPlayer.cpp File Reference

```
#include "HumanPlayer.h"
```

PROItheGame/HumanPlayer.h File Reference

```
#include "Player.h"
```

Classes

- class **HumanPlayer**

PROItheGame/Macros.h File Reference

Macros

- #define **ID_DEV_LEVEL_1** 1
- #define **ID_DEV_LEVEL_2** 2
- #define **ID_DEV_LEVEL_3** 3
- #define **ID_DEV_LEVEL_4** 4
- #define **ID_EASY_LEVEL_1** 101
- #define **ID_EASY_LEVEL_2** 102
- #define **ID_EASY_LEVEL_3** 103
- #define **ID_EASY_LEVEL_4** 104

Macro Definition Documentation

#define ID_DEV_LEVEL_1 1

Definition at line 4 of file Macros.h.

#define ID_DEV_LEVEL_2 2

Definition at line 6 of file Macros.h.

#define ID_DEV_LEVEL_3 3

Definition at line 8 of file Macros.h.

#define ID_DEV_LEVEL_4 4

Definition at line 10 of file Macros.h.

#define ID_EASY_LEVEL_1 101

Definition at line 12 of file Macros.h.

#define ID_EASY_LEVEL_2 102

Definition at line 14 of file Macros.h.

#define ID_EASY_LEVEL_3 103

Definition at line 16 of file Macros.h.

#define ID_EASY_LEVEL_4 104

Definition at line 18 of file Macros.h.

PROItheGame/MenuObject.cpp File Reference

```
#include "MenuObject.h"
```

PROItheGame/MenuObject.h File Reference

```
#include <string>
#include <SDL.h>
#include <SDL_ttf.h>
#include "GameObject.h"
```

Classes

- class **MenuObject**

PROItheGame/Momentum.h File Reference

Classes

- class **Momentum< T1, T2, T3 >**

PROItheGame/Obstacle.cpp File Reference

```
#include "Obstacle.h"
```

PROItheGame/Obstacle.h File Reference

```
#include "GameObject.h"
```

Classes

- class **Obstacle**

PROItheGame/Player.cpp File Reference

```
#include "Player.h"
```

PROItheGame/Player.h File Reference

```
#include <vector>
#include "Obstacle.h"
#include "Momentum.h"
```

Classes

- class **Player**

PROItheGame/PROItheGame.cpp File Reference

```
#include <SDL.h>
#include <iostream>
#include "Game.h"
```

Functions

- int **main** ()

Function Documentation

int main ()

Definition at line 15 of file PROItheGame.cpp.

PROItheGame/Tree.cpp File Reference

PROItheGame/Tree.h File Reference

```
#include "TreeElement.h"
```

Classes

- class **Tree< T >**

PROItheGame/TreeElement.cpp File Reference

PROItheGame/TreeElement.h File Reference

```
#include <vector>
```

Classes

- class **TreeElement< T >**

UnitTestProject/Test.cpp File Reference

```
#include "Test.h"
```

UnitTestProject/Test.h File Reference

```
#include <iostream>
#include <assert.h>
#include "../PROItheGame/Game.h"
```

Classes

- class **Test**

UnitTestProject/UnitTestProject.cpp File Reference

```
#include "Test.h"
```

Functions

- int **main** ()

Function Documentation

int main ()

Definition at line 14 of file UnitTestProject.cpp.

Index