435Lab Project Game

Generated by Doxygen 1.8.6

Sun Apr 15 2018 15:30:45

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

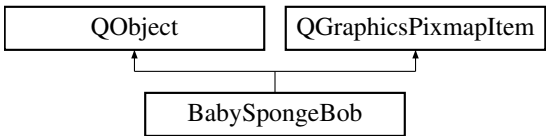
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 BabySpongeBob Class Reference

Inheritance diagram for BabySpongeBob:



Public Slots

- void update ()

  updates c

Public Member Functions

- BabySpongeBob (int healthyItemsFed=0, QObject *parent=nullptr)

Public Attributes

- QTimer * **timer**
- int healthyItemsFed

  this variable stores the number of healthy items fed to the baby

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BabySpongeBob::BabySpongeBob (int *healthyItemsFed* = `0`, QObject * *parent* = `nullptr`) [explicit]

< setting picture of baby

< starting timer

< setting number of healthy items already fed (in case of pause)

< timer for update

4.1.2 Member Function Documentation

4.1.2.1 void BabySpongeBob::update ( ) [slot]

updates c

BabySpongeBob::update, updates the game metrics in case an item reaches the baby. if the item is of type grabbable, the function reachedBaby is called to update the metrics

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

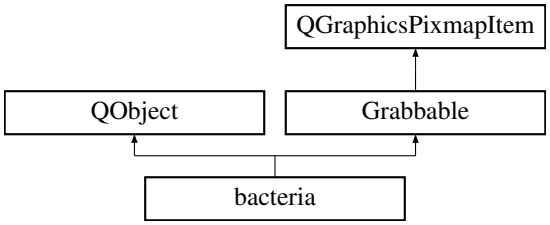
- babyspongebob.h
- babyspongebob.cpp

4.2 bacteria Class Reference

bacteria class

`#include <bacteria.h>`

Inheritance diagram for bacteria:



Public Slots

- void update ()

  update the location on the screen and detects collisions

Public Member Functions

- bacteria (int strength, int direction, int directionY, double Xvelocity, double Yvelocity, int deviationLimit, int centerline, Header *header=NULL, QString game="", QObject *parent=nullptr)
- ~bacteria ()

  destructor

Public Attributes

- QTimer * timer

  timer attribute that specifies the timer
- int direction

  direction attribute that specifies the direction of movement of the bacteria
- int directionY

  attribute that specifies the Y direction of movement of the virus
- int Xvelocity

  attribute that specifies the X velocity of the virus
- int Yvelocity

attribute that specifies the X velocity of the virus

- int deviationLimit

  specifies maximum deviation from center line

- int centerline

  specifies the center of propagation of the virus

- int **upperlimit**
- QString **game**

4.2.1 Detailed Description

bacteria class

.h A bacteria is an element on screen that moves periodically in a predefined direction.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 bacteria::bacteria (int *strength,* int *direction,* int *directionY,* double *Xvelocity,* double *Yvelocity,* int *deviationLimit,* int *centerline,* Header * *header* = NULL, QString *game* = " ", QObject * *parent* = nullptr) [explicit]

setting attributes

starting timer and connecting it

4.2.3 Member Function Documentation

4.2.3.1 void bacteria::update ( ) [slot]

update the location on the screen and detects collisions

bacteria::update updated the position of the bacteria. It also checks if theres is a collision with any object and checks its type. additionally, it updates the metrics if there is a collision with the player (spongebob) checks if the item is at the boundary of the screen to remove it

checks if there is a collision

checks if the collision is with the player

<checking if the player isnt strong enough to kill the bacteria, if true, he should lose a life.

update lives, deletes item,updates cleanliness, and reset stats

< player is strong enough to kill the bacteria

deletes item,updates cleanliness, and updates stats

updating item position if the player has the followme tag toggled on, this means that the bacteria should follow him. an algorithm is used to calculate the speed and direction at which the item should move to follow the palyer.

else, the bacteria moves in a wave motion along a fixed center line form the left to the right or the opposite based on its direction.

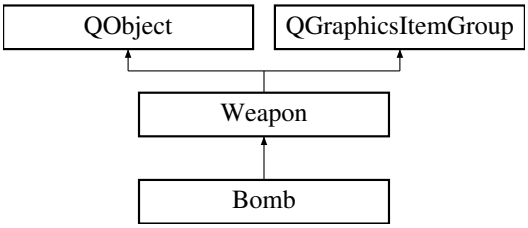
algorithm to follow the plater

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

- bacteria.h
- bacteria.cpp

4.3 Bomb Class Reference

Inheritance diagram for Bomb:



Public Slots

- void update ()

  Bomb::update updates the position of the bomb and checks for collision.

Public Member Functions

- Bomb (int strength)

  Bomb::Bomb constructor.

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Bomb::Bomb (int *strength*)

Bomb::Bomb constructor.

Parameters

strength	the strength of the bomb (how fast it goes)

setting attributes

4.3.2 Member Function Documentation

4.3.2.1 void Bomb::update ( ) [slot]

Bomb::update updates the position of the bomb and checks for collision.

if not ready to throw the bomb, setting it.

< adding number of steps passed.

there is a collision with an item that is grabbable,there are items to delete.

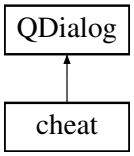
number of teps exceeds limit. bomb is deleted

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

- bomb.h
- bomb.cpp

4.4 cheat Class Reference

Inheritance diagram for cheat:



Public Member Functions

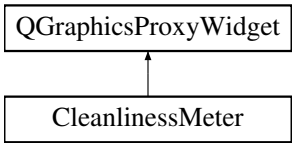
- **cheat** (QWidget ∗parent=0)

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

- cheat.h
- cheat.cpp

4.5 CleanlinessMeter Class Reference

Inheritance diagram for CleanlinessMeter:



Public Member Functions

- void **UpdateValue** ()

Public Attributes

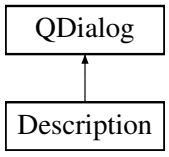
- QProgressBar ∗ **ProgressBar**

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

- cleanlinessmeter.h
- cleanlinessmeter.cpp

4.6 Description Class Reference

Inheritance diagram for Description:



Public Member Functions

- **Description** (QWidget ∗parent=0, QString game="")

Public Attributes

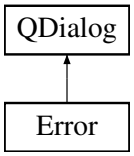
- QString **Game**

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

- description.h
- description.cpp

4.7 Error Class Reference

Inheritance diagram for Error:



Public Member Functions

- **Error** (QString msg, QWidget ∗parent=0)

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

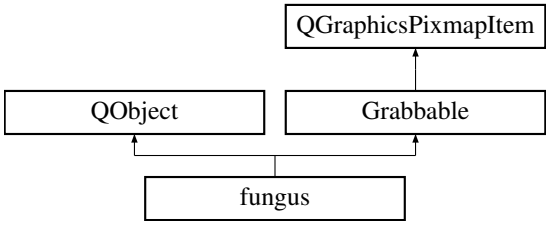
- error.h
- error.cpp

4.8 fungus Class Reference

fungus class

`#include <fungus.h>`

Inheritance diagram for fungus:



Public Slots

- void update ()

  update the location on the screen and detects collisions

Public Member Functions

- fungus (Header *header, QString name="", QObject *parent=nullptr)

  fungus::fungus

Public Attributes

- QTimer * timer

  timer attribute that specifies the timer
- int Xvelocity

  y velocity of fungus
- int Yvelocity

  x velocity of fungus
- int timetolive

  specifies time left to die
- QString **game**

4.8.1 Detailed Description

fungus class

.h A fungus is an element on screen that moves periodically following spongebob.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 fungus::fungus (Header * *header*, QString *game* = " ", QObject * *parent* = `nullptr`) [explicit]

fungus::fungus

Parameters

header	
parent	constructor of a fungus, creates an instance and provides is with a velocity, time to live, and a pointer to the header

4.8.3 Member Function Documentation

4.8.3.1 void fungus::update ( ) [slot]

update the location on the screen and detects collisions

fungus::update

updates the position, detects collision, and in case of collision with the player, it penalises him it also deletes the fungus if its time is over collidelist this creates a list of all colliding items. it then dynamically casts each detected item to an instance of spongebob if any item returns true which means that it exists, this means that a collision with spongebob has occured this leads to some procedures

immunity is halved

cleanliness is halved

instructs batceria to follow him

timer for following starts

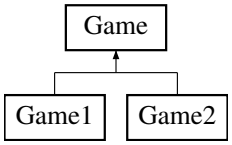
this is an algorithm to detect the position of spongebob and follow him

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

- fungus.h
- fungus.cpp

4.9 Game Class Reference

Inheritance diagram for Game:



Public Types

- enum **GameDifficulty** { **easy** = 1, **medium** = 2, **hard** = 3 }
- enum **GameMode** {
  New = 1, **Resume** = 2, **Over** = 3, **Win** = 4,
  Pause = 5 }

Public Member Functions

- void **SetDifficulty** (bool easyRadio, bool mediumRadio, bool hardRadio)

Public Attributes

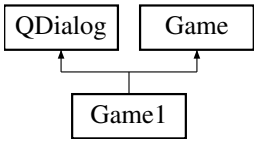
- User ∗ **user**
- int **Difficulty**
- GameView ∗ **gameView**

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

- game.h
- game.cpp

4.10 Game1 Class Reference

Inheritance diagram for Game1:



Public Member Functions

- Game1 (QWidget ∗parent=0, User ∗user=new User())

Game1::Game1 constructor for the class. it also sets the radio buttons to enabled or disabled based on the players previous records.

- ~Game1 ()

  distructor

Static Public Attributes

- static QString **name** = "Game1"

Additional Inherited Members

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Game1::Game1 (QWidget * *parent* = 0, User * *user* = new User()) [explicit]

Game1::Game1 constructor for the class. it also sets the radio buttons to enabled or disabled based on the players previous records.

Parameters

parent	
user	

getting previous records

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

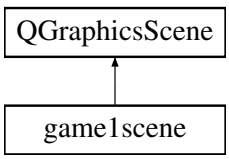
- game1.h
- game1.cpp

4.11 game1scene Class Reference

game1scene class

`#include <game1scene.h>`

Inheritance diagram for game1scene:



Public Slots

- void addhuItems ()

  add huItems on the screen

- void addbacteria ()

  add bacteria on the screen

- void addvirus ()

  add viruses on the screen

- void addfungus ()

  add fungii on the screen

- void CloseView ()

  game1scene::CloseView

Public Member Functions

- game1scene (GameView *gameView, int gameMode, QString username, int difficulty=1, Header *header=nullptr, bool paused=false)

  constructor
- void GameOver ()

  game1scene::GameOver
- void WonGame ()

  game1scene::WonGame
- void PauseGame ()

  game1scene::PauseGame
- void **keyPressEvent** (QKeyEvent *event)
- void keyReleaseEvent (QKeyEvent *event)

  Detects key release events.
- void **mouseReleaseEvent** (QGraphicsSceneMouseEvent *event)

Public Attributes

- Header * header

  pointer to header
- QTimer * addhuItemstimer

  QTimer attribute,.
- QTimer * addbacteriatimer

  QTimer attribute,.
- QTimer * addvirustimer

  QTimer attribute,.
- QTimer * followtimer

  QTimer attribute,.
- QTimer * addfungustimer

  QTimer attribute,.
- GameView * **gameView**
- bool **paused**
- bool **completed**

4.11.1 Detailed Description

game1scene class

.h This creates an instance of game1scene which includes a spongebob and other items.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 game1scene::game1scene (GameView * *gameView,* int *gameMode,* QString *username,* int *difficulty* = `1`, Header * *header* = `nullptr`, bool *paused* = `false`) [explicit]

constructor

game1scene::game1scene constructs the game1scene and all of its attributes, starts the timers, and connects the signal with its slot player has won the game

the game is being paused

this is a new game

this is a game that was paused and is now being resumed

setting all atributes as they were previously

4.11.3 Member Function Documentation

4.11.3.1 void game1scene::addbacteria ( ) [slot]

add bacteria on the screen

game1scene::addbacteria

first we check if the cleanness is less than 100 if true, we continue with the procedure of adding a bacteria to the scene a random number is first choosen between 0 and 3 a bacteria is created if the random number is less than 2, the direction is set from left to right else it is set from right to left

the position is calculated by the following equation : 50 + randomnumber $*$100 so the results can be: 50, 150, 250, 350

the strength of the bacteria is also generated randomly

velocity is generated randomly with an influance of the difficulty of the game the harder the game, the faster the bacteria $<$ this variable is used to store the power of the bacteria that can still be added to the scene.d

4.11.3.2 void game1scene::addfungus ( ) [slot]

add fungii on the screen

game1scene::addfungus

a random number is first choosen for the x and y positions then the distance to the player is calculated if the distance is less than 250, the position is randomly generated again

when the distance criteria is met, an instance of the fungus is created with the generated coordinates

4.11.3.3 void game1scene::addhultems ( ) [slot]

add hultems on the screen

game1scene::addhultems

creates new hultems with a random direction and starting position a random number is first choosen between 0 and 20 a hultem instance is created if the random number is less than 2, the direction is set from left to right else it is set from right to left

the position is calculated by tho following equation : 50 + randomnumber $*$100 so the results can be: 50, 150, 250, 350

4.11.3.4 void game1scene::addvirus ( ) [slot]

add viruses on the screen

game1scene::addvirus

this is a function to add a virus to the screen a random number is first choosen between 0 and 1 to set direction the starting y posiion is then chosen randomly a virus is created

4.11.3.5 void game1scene::CloseView ( ) [slot]

game1scene::CloseView

this function is exited closing the window.

4.11.3.6 void game1scene::GameOver ( )

game1scene::GameOver

if user loses, the game is over.

4.11.3.7 void game1scene::keyReleaseEvent (QKeyEvent * *event*)

Detects key release events.

game1scene::keyReleaseEvent

Parameters

**event*	first argument, key release event
event	this collects key release events and removes them from the set of pressed keys

4.11.3.8 void game1scene::PauseGame ( )

game1scene::PauseGame

this function is called when pausing the game is needed. the header and player stats are saved in the files.

4.11.3.9 void game1scene::WonGame ( )

game1scene::WonGame

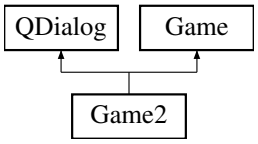
this function is exited when a player wins the game. their score is saved, and the scene is closed.

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

- game1scene.h
- game1scene.cpp

4.12 Game2 Class Reference

Inheritance diagram for Game2:



Public Member Functions

- **Game2** (QWidget *parent=0, User *user=new User())
- ~Game2 ()

  destructor

Static Public Attributes

- static QString **name** = "Game2"

Additional Inherited Members

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

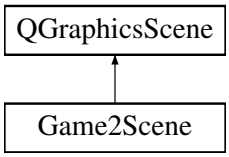
- game2.h
- game2.cpp

4.13 Game2Scene Class Reference

Game2Scene class.

`#include <game2scene.h>`

Inheritance diagram for Game2Scene:



Public Slots

- void addhuItems ()

  add huItems on the screen
- void **CloseView** ()

Public Member Functions

- Game2Scene (GameView *gameView, int gameMode, QString username, int difficulty=1, Header *header=nullptr, bool paused=false)

  constructor
- void GameOver ()

  Game2Scene::GameOver this is excited when a player looses the game.
- void WonGame ()

  Game2Scene::WonGame this is excited when a player wins the game.
- void PauseGame ()

  Game2Scene::PauseGame this is excited when a player pauses the game.
- void **keyPressEvent** (QKeyEvent *event)
- void keyReleaseEvent (QKeyEvent *event)

  Detects key release events.
- void mouseReleaseEvent (QGraphicsSceneMouseEvent *event)

  Game2Scene::mouseReleaseEvent.
- ~Game2Scene ()

  destructor

Public Attributes

- Header * header

  pointer to header
- QTimer * addItemToQueueTimer

QTimer attribute,.

- GameView * **gameView**
- bool **paused**
- bool **completed**

4.13.1 Detailed Description

Game2Scene class.

.h This creates an instance of Game2Scene which includes a spongebob and other items.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Game2Scene::Game2Scene (GameView * *gameView,* int *gameMode,* QString *username,* int *difficulty* = `1`, Header * *header* = `nullptr`, bool *paused* = `false`) [explicit]

constructor

Game2Scene::Game2Scene constructs the Game2Scene and all of its attributes, starts the timers, and connects the signal with its slot. player lost the game

player won the game

the game is being paused

starting a new game

resuming from a paused game

4.13.3 Member Function Documentation

4.13.3.1 void Game2Scene::addhuItems ( ) [slot]

add huItems on the screen

Game2Scene::addhuItems this function is responsible for adding healthy/unhealthy items to the screen.

healty/unhealthy ratio is determined by the gmae difficulty where a ratio of 3:1 is used for the easy level, and a 1:3 is used for the hard one the image of the healthy or unhealthy item is also spceified randomly from 4 pictures for each type. first number (0 or1) is for type of item (1 is healthy and 0 is unhealthy) the second is for the picture to display

4.13.3.2 void Game2Scene::keyReleaseEvent (QKeyEvent * *event*)

Detects key release events.

Game2Scene::keyReleaseEvent.

Parameters

**event*	first argument, key release event
event	this collects key release events and removes them from the set of pressed keys

4.13.3.3 void Game2Scene::mouseReleaseEvent (QGraphicsSceneMouseEvent * *event*)

Game2Scene::mouseReleaseEvent.

Parameters

event	this detects the mouse release event in order to pause the game

4.13.3.4 void Game2Scene::PauseGame ( )

Game2Scene::PauseGame this is excited when a player pauses the game.

the player metrics (score, timer, number of lives...) are saved to be retreived on the resume. if the player is a registered one, his metrics are also stored in the game database for later retreval. else wise, his metrics are saved as long as the game window is not closed.

4.13.3.5 void Game2Scene::WonGame ( )

Game2Scene::WonGame this is excited when a player wins the game.

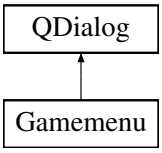
if the player is a registered one (not a guest), his score is added to his history of scores. additionally, winning a game would allow the user to play a higher level.

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

- game2scene.h
- game2scene.cpp

4.14 Gamemenu Class Reference

Inheritance diagram for Gamemenu:



Public Member Functions

- **Gamemenu** (QWidget ∗parent=0, QString game="", User ∗user=new User())
- ∼Gamemenu ()

  destructor

Public Attributes

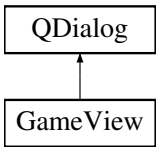
- QString **Game**
- User ∗ **user**

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

- gamemenu.h
- gamemenu.cpp

4.15 GameView Class Reference

Inheritance diagram for GameView:



Public Member Functions

- **GameView** (QWidget ∗parent=0)
- void setScene (QGraphicsScene ∗gameScene)

  creating the scene (game 1 or 2), and centering the game in the middle of the screen.

Public Attributes

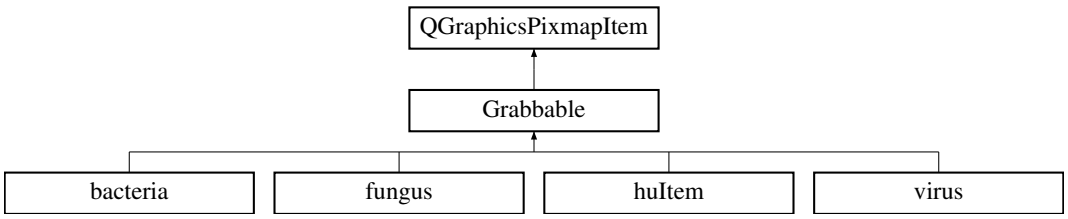
- QGraphicsScene ∗ **gameScene**

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

- gameview.h
- gameview.cpp

4.16 Grabbable Class Reference

Inheritance diagram for Grabbable:



Public Member Functions

- void wasGrabbed ()

  Grabbable::wasGrabbed this function is excited when an item is hooked.
- void wasShot (Weapon ∗by)

  excited when an item is shot (laser or bomb)
- void reachedBaby ()

  Grabbable::reachedBaby this is a function that is excited when the baby collides with an item.

Public Attributes

- Header * header
  pointer to the header
- int **strength**
- int **interval**
- bool **grabbed**

4.16.1 Member Function Documentation

4.16.1.1 void Grabbable::reachedBaby ( )

Grabbable::reachedBaby this is a function that is excited when the baby collides with an item.

checking if the item collided with is an huitem.

if the huitem is an unhealthy item, the player loses a life.

if the huitem is a healthy item, the cleanliness increases.

removing the item that collided with the baby.

4.16.1.2 void Grabbable::wasGrabbed ( )

Grabbable::wasGrabbed this function is excited when an item is hooked.

adding or subtracting time and immunity based on the type of the grabbed item

4.16.1.3 void Grabbable::wasShot (Weapon * *by*)

excited when an item is shot (laser or bomb)

if the item collides with a bomb, it loses all of its strength

if the item was shot with a laser, its strength decreases by the strength of the laser

if this is a healthy item, increase score.

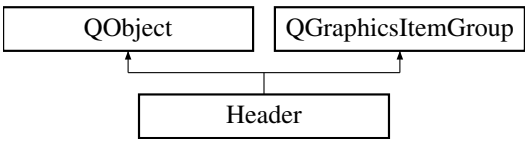
if this is an unhealthy item, decrease score.

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

- grabbable.h
- grabbable.cpp

4.17 Header Class Reference

Inheritance diagram for Header:



Public Slots

- void **CountDown** ()

Public Member Functions

- Header (SpongeBob *player, int difficulty, QString username, QString game, bool paused, int time, int healthyItemsFed=0)
- void SetCleanliness (int val)

  Header::SetCleanliness.
- void SetImmunity (int val)

  Header::SetImmunity.
- void SetScore (int val)
- void **SetTime** (int val)
- void **RemoveLife** ()
- void **RemoveBomb** ()
- void AddCleanlMeter (int x, int y)

  Header::AddCleanlMeter.
- void **AddChart** (int x, int y, int width, int height, int startAngle, int spanAngle, QColor color)
- void **AddHearts** (int x, int y)
- void **AddTime** (int x, int y)
- void **AddPause** (int x, int y)
- void **AddLevel** (int x, int y)
- void **AddScore** (int x, int y)
- void AddNeedle (int x, int y)

  Header::AddNeedle.
- void **AddBaby** (int x, int y, int healthyItemsFed)
- void AddBombs (int x, int y)

  Header::AddBombs.
- void **AddScoreCalculation** (int x, int y)
- void Render ()

  Header::Render this is the main function for the visual representation of the updates of the metrics.

Public Attributes

- CleanlinessMeter * **cleanlinessMeter**
- Pause * **pause**
- QGraphicsPixmapItem * **hearts** [3]
- QGraphicsPixmapItem * **bombs** [3]
- QGraphicsTextItem * **timeLabel**
- QGraphicsTextItem * **levelLabel**
- QGraphicsTextItem * **scoreLabel**
- QGraphicsTextItem * **scoreCalculationLabel**
- QGraphicsLineItem * **needle**
- SpongeBob * **player**
- BabySpongeBob * **baby**
- QString **username**
- QString **game**
- QTimer * **timer**
- int **time**
- int **difficulty**
- int **currentBacteriaCountInScene**
- int **counter**
- bool **paused**

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Header::Header (SpongeBob * *player*, int *difficulty*, QString *username*, QString *game*, bool *paused*, int *time*, int *healthyItemsFed* = 0)

setting metrics

setting graphics

4.17.2 Member Function Documentation

4.17.2.1 void Header::AddBombs (int *x*, int *y*)

Header::AddBombs.

Parameters

x	
y	this function adds bombs to the header of the game scene

4.17.2.2 void Header::AddCleanlMeter (int *x*, int *y*)

Header::AddCleanlMeter.

Parameters

x	
y	adds the cleanliness meter to the scene, the position of the meter is set by the x and y coordinates

4.17.2.3 void Header::AddNeedle (int *x*, int *y*)

Header::AddNeedle.

Parameters

x	
y	this function adds the needle to the chart in the header of the game. the needle acts as an indicator to the strength of the player

4.17.2.4 void Header::Render ( )

Header::Render this is the main function for the visual representation of the updates of the metrics.

it is called when an update to a metric occures, and it updates accordingly the visual representation of the metrics in the header

4.17.2.5 void Header::SetCleanliness (int *val*)

Header::SetCleanliness.

Parameters

val	the val is added(or subtracted if negative) from the total cleanliness level if the level is less than 0 or greater than 100, the value is reseted to zero or 100

4.17.2.6 void Header::SetImmunity (int *val*)

Header::SetImmunity.

Parameters

val	this sets the immunity by adding(or subtracting) the value of val from the current immmunity level

4.17.2.7 void Header::SetScore (int *val*)

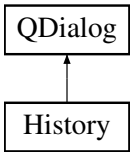
< this visually illustrates adding the score.

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

- header.h
- header.cpp

4.18 History Class Reference

Inheritance diagram for History:



Public Member Functions

- History (QWidget *parent=0, User *user=new User())
- ~History ()

  destructor

Public Attributes

- User * **user**

4.18.1 Constructor & Destructor Documentation

4.18.1.1 History::History (QWidget * *parent* = `0`, User * *user* = `new` User `()`) `[explicit]`

< displays highest score for the user for game1

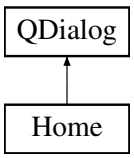
< displays highest score for the user for game2

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

- history.h
- history.cpp

4.19 Home Class Reference

Inheritance diagram for Home:



Public Member Functions

- Home (QWidget *parent=0, User *user=new User())
- ~Home ()

  destructor

Public Attributes

- User * **user**

4.19.1 Constructor & Destructor Documentation

4.19.1.1 Home::Home (QWidget * *parent* = `0`, User * *user* = `new` User `()`) `[explicit]`

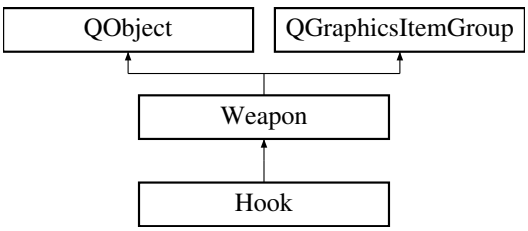
< happy birthday if his birthday is today

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

- home.h
- home.cpp

4.20 Hook Class Reference

Inheritance diagram for Hook:



Public Slots

- void update ()

  Hook::update the update function updates the length of the rope of the hook for a better graphical experience.

Public Member Functions

- Hook (int strength)

  Hook::Hook.

Additional Inherited Members

4.20.1 Constructor & Destructor Documentation

4.20.1.1 Hook::Hook (int *strength*)

Hook::Hook.

constructor

Parameters

strength	used to specify the speed of movement of the hook. as the player gets stronger, the hook is faster

initializing

drawing the rope

drawing the head of the hook

< connecting the timer to an update function that updates the length of the hook

4.20.2 Member Function Documentation

4.20.2.1 void Hook::update ( ) [slot]

Hook::update the update function updates the length of the rope of the hook for a better graphical experience.

< adding length to the rope

< updating the position of the head

if the rope has reached its minimum length

if the rope has reached its maximum length

if it hasnt caught any item yet

checking for collisions

checking if the collision is with a grabbable item

updating attributes

in case it caught an item

< positioning the item on the head of the hook

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

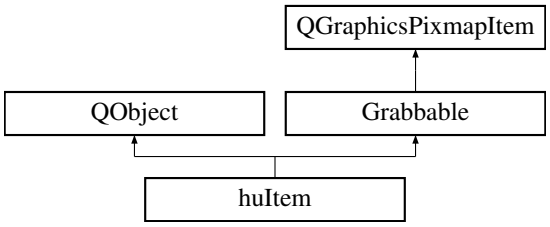
- hook.h
- hook.cpp

4.21 huItem Class Reference

huItem class

`#include <huItem.h>`

Inheritance diagram for huItem:



Public Slots

- void update ()

  update the huItemss location on the screen

Public Member Functions

- **huItem** (bool type, Header *header, QString game="", QObject *parent=nullptr, int strength=0, int interval=100)
- ~huItem ()

  destructor

Public Attributes

- QTimer * timer

  timer attribute that specifies the timer
- bool **type**
- QString **game**

4.21.1 Detailed Description

huItem class

.h A huItems is an element on screen that moves periodically in a predefined direction.

4.21.2 Member Function Documentation

4.21.2.1 void huItem::update ( ) [slot]

update the huItemss location on the screen

huItem::update

constantly updates the movement of the item in a downwards motion for game 1, and in an ellipse motion for game 2. for game 1: If the item collides with spongebob, his health is updated and the item is removed from the scene. If it collides with other items, it is not affected. additionally, when the item reaches the boundary of the scene, it is deleted.

for game 2: the item can be hooked or fired. if the item reaches the baby, based on its type, it might affect the babies health positivly or negatively. game 1

game 2

item is not grabbed yet

moves along a straigth downwards path first

then through an ellipse

then through a straight upwards path

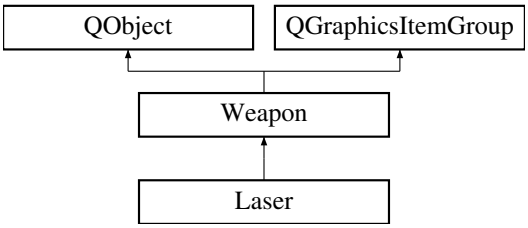
removed from the scene if it has exceeded the defined path

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

- huItem.h
- huItem.cpp

4.22 Laser Class Reference

Inheritance diagram for Laser:



Public Slots

- void update ()

  Laser::update this function updates the length of the laser while shooting for a better user experience. It also checks for collisions and acts accordingly by calling other functions.

Public Member Functions

- Laser (int strength)

  constructor

Additional Inherited Members

4.22.1 Member Function Documentation

4.22.1.1 void Laser::update ( ) [slot]

Laser::update this function updates the length of the laser while shooting for a better user experience. It also checks for collisions and acts accordingly by calling other functions.

check if there is a collison with a grabbable item (huitem)

< stop elongating the laser

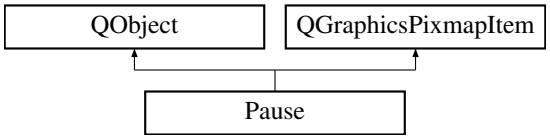
< calling the wasshot function for updating metrics and removing the item from the scene

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

- laser.h
- laser.cpp

4.23 Pause Class Reference

Inheritance diagram for Pause:



Public Member Functions

- **Pause** (QObject *parent=nullptr)
- void **mouseReleaseEvent** (QGraphicsSceneMouseEvent *event)

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

- pause.h
- pause.cpp

4.24 Scores Class Reference

The Scores class.

`#include <scores.h>`

Static Public Member Functions

- static QString GetHighestScore (QString game)

  Scores::GetHighestScore.
- static QStringList GetUserScores (QString username, QString game)

  Scores::GetUserScores.
- static bool AddScore (QString username, QString score, QString game)

  Scores::AddScore.

4.24.1 Detailed Description

The Scores class.

this class containes everything related to scores and manipulating them

4.24.2 Member Function Documentation

4.24.2.1 bool Scores::AddScore (QString *username,* QString *score,* QString *game*) [static]

Scores::AddScore.

Parameters

username	
score	
game	

Returns

this function adds a score to the list of scores for a selected user in a selected game

4.24.2.2 QString Scores::GetHighestScore (QString *game*) [static]

Scores::GetHighestScore.

Parameters

game	

Returns

this function gets the highest score in a given game from a Json Document

4.24.2.3 QStringList Scores::GetUserScores (QString *username*, QString *game*) [static]

Scores::GetUserScores.

Parameters

username	
game	

Returns

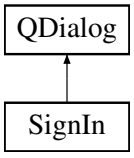
this function gets the scores of the user from a Json document

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

- scores.h
- scores.cpp

4.25 SignIn Class Reference

Inheritance diagram for SignIn:



Public Member Functions

- SignIn (QWidget ∗parent=0)
  SignIn::SignIn.
- ∼SignIn ()
  desctructor

4.25.1 Constructor & Destructor Documentation

4.25.1.1 SignIn::SignIn (QWidget ∗ *parent* = 0) [explicit]

SignIn::SignIn.

Parameters

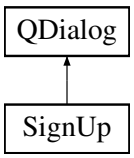
parent	constructor

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

- signin.h
- signin.cpp

4.26 SignUp Class Reference

Inheritance diagram for SignUp:



Public Member Functions

- SignUp (QWidget ∗parent=0)
  SignUp::SignUp.
- ∼SignUp ()
  destructor

Public Attributes

- QString **profilePictureEdit**

4.26.1 Constructor & Destructor Documentation

4.26.1.1 SignUp::SignUp (QWidget ∗ *parent* = 0) [explicit]

SignUp::SignUp.

Parameters

parent	constructor for the signup class. it hides the validation error messages

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

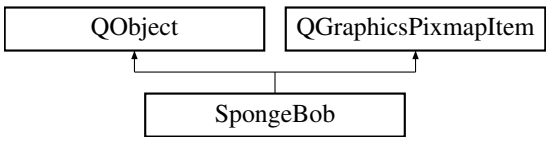
- signup.h
- signup.cpp

4.27 SpongeBob Class Reference

spongeBob class

`#include <spongeBob.h>`

Inheritance diagram for SpongeBob:



Public Slots

- void toggleFollow ()

  change from follow me to dont follow me

Public Member Functions

- SpongeBob (int cleanliness, int immunity, int lives, int score, QPoint pos, QString game, QObject ∗parent=nullptr, int bombs=0, int requiredBombScore=0, int translation=0, QString weapon="", int weaponStrength=0)

  constructor
- void keyPressEvent (QKeyEvent ∗event)

  Detects key strokes pressed.
- void keyReleaseEvent (QKeyEvent ∗event)

  Detects key release events.

Public Attributes

- bool followme

  followme attribute that specifies if the bacteria should follow spongebob or not
- QTimer ∗ followTimer

  timer attribute that specifies the timer
- int cleanliness

  cleanliness of the tank
- int immunity

  immunity level of spongebob
- int lives

  number of lives
- int score

total score

- QPoint currentPos

  current position

- Weapon * weapon

  pointer to the weapon used

- QString game

  specify the game to which the instance belongs (game1 or game2)

- int **translation**
- int **bombs**
- int **requiredBombScore**

4.27.1 Detailed Description

spongeBob class

.h This creates an instance of the spongebob.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 SpongeBob::SpongeBob (int *cleanliness,* int *immunity,* int *lives,* int *score,* QPoint *pos,* QString *game,* QObject * *parent* = `nullptr`, int *bombs* = `0`, int *requiredBombScore* = `0`, int *translation* = `0`, QString *weapon* = " ", int *weaponStrength* = `0`) [explicit]

constructor

setting up attributes of spongebob

4.27.3 Member Function Documentation

4.27.3.1 void SpongeBob::keyPressEvent (QKeyEvent * *event*)

Detects key strokes pressed.

spongeBob::keyPressEvent

Parameters

**event*	first argument, keystroke event
event	the keypress event detects all key strokes on the keyboard. we are only interested in the up,down,left, and right strokes when one of those keys is pressed, spongebob moves accordingly when possible. it also moves diagonally if multiple of these keys are pressed at the same time. Sponge bob moves diagonally faster than horizontally and vertically if the location of spongebob is on the edges of the screen, the position isnt updated if this will result in leaving the screen

game 1 movement

the x key fires the weapon

the z key changes the weapon

if the player has the required score and has bombs

4.27.3.2 void SpongeBob::keyReleaseEvent (QKeyEvent * *event*)

Detects key release events.

spongeBob::keyReleaseEvent

Parameters

∗event	first argument, key release event
event	this collects key release events and removes them from the set of pressed keys

4.27.3.3 void SpongeBob::toggleFollow ( ) [slot]

change from follow me to dont follow me

spongeBob::toggleFollow toggles the follow me to false when the timer ends

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

- spongeBob.h
- spongeBob.cpp

4.28 User Class Reference

Public Member Functions

- User (QString username, QString password, QString firstName, QString lastName, QString dateOfBirth, QString gender, QImage profilePicture)

Static Public Member Functions

- static bool AddUser (User user)

  User::AddUser.
- static User ∗ GetUser (QString username, QString password)

  User::GetUser.
- static QJsonObject UserToJson (User user)

  User::UserToJson.
- static User ∗ JsonToUser (QJsonObject object, QString username)

  User::JsonToUser.
- static void PauseGameForUser (Header ∗header, bool completed)
- static Header ∗ ResumeGameForUser (QString game, QString username)
- static int GetUserLevel (QString game, QString username)

  this function is used to get the current level of the user in a specific game
- static void UpgradeUserToLevel (QString game, QString username, int level)

  User::UpgradeUserToLevel.

Public Attributes

- QString **Username**
- QString **Password**
- QString **FirstName**
- QString **LastName**
- QString **DateOfBirth**
- QString **Gender**
- QImage **ProfilePicture**

4.28.1 Constructor & Destructor Documentation

4.28.1.1 User::User (QString *username,* QString *password,* QString *firstName,* QString *lastName,* QString *dateOfBirth,* QString *gender,* QImage *profilePicture*) [explicit]

setting attributes

4.28.2 Member Function Documentation

4.28.2.1 bool User::AddUser (User *user*) [static]

User::AddUser.

Parameters

user	

Returns

this function adds a new user and saves the records to the json file

4.28.2.2 User * User::GetUser (QString *username,* QString *password*) [static]

User::GetUser.

Parameters

username	
password	

Returns

this function searches for a user based on a username and a password

4.28.2.3 User * User::JsonToUser (QJsonObject *object,* QString *username*) [static]

User::JsonToUser.

Parameters

object	
username	

Returns

this function takes a json object and parses it to an user object

4.28.2.4 void User::PauseGameForUser (Header * *header,* bool *completed*) [static]

this function is used to pause the game for later access. it saves the data in a json document. the values of attributes in the header are saved.

4.28.2.5 Header * User::ResumeGameForUser (QString *game,* QString *username*) [static]

this function is given a game and a username, and is used to check if there are available paused games for the user. if there is a paused game, an instance of header and spongebob are created and filled with the previously saved state. initializing according to previous metrics

initializing according to previous metrics

initializing according to previous metrics

initializing according to previous metrics

4.28.2.6 void User::UpgradeUserToLevel (QString *game,* QString *username,* int *level*) [static]

User::UpgradeUserToLevel.

Parameters

game	
username	
level	this function is called whith a game, a username, and a level. it gets the json object of the user using its username, and then assigns him a higher level on the given game. this allows the user to play the game on a higher level.

4.28.2.7 QJsonObject User::UserToJson (User *user*) [static]

User::UserToJson.

Parameters

user	

Returns

this function takes an instance of a user and converts its attributes to a json object

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

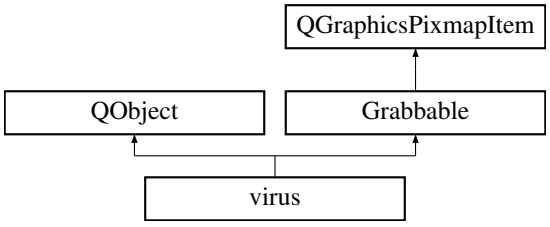
- user.h
- user.cpp

4.29 virus Class Reference

virus class

`#include <virus.h>`

Inheritance diagram for virus:



Public Slots

- void update ()

  update the location on the screen and detects collision

Public Member Functions

- virus (int direction, int directionY, int Xvelocity, int Yvelocity, int deviationLimit, int centerline, Header *header, QString game="", QObject *parent=nullptr)

  virus::virus

Public Attributes

- QTimer * timer

  timer attribute that specifies the timer
- int direction

  attribute that specifies the X direction of movement of the virus
- int directionY

  attribute that specifies the Y direction of movement of the virus
- int Xvelocity

  attribute that specifies the X velocity of the virus
- int Yvelocity

  attribute that specifies the X velocity of the virus
- int deviationLimit

  specifies maximum deviation from center line
- int centerline

  specifies the center of propagation of the virus
- int **foobar**
- QString **game**

4.29.1 Detailed Description

virus class

.h A virus is an element on screen that moves periodically in a predefined direction.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 virus::virus (int *direction*, int *directionY*, int *Xvelocity*, int *Yvelocity*, int *foobar*, int *centerline*, Header * *header*, QString *game* = " ", QObject * *parent* = `nullptr`) [explicit]

virus::virus

Parameters

direction	
directionY	
Xvelocity	
Yvelocity	

foobar	
centerline	
player	
parent	constructor

for periodic update of the position

4.29.3 Member Function Documentation

4.29.3.1 void virus::update ( ) [slot]

update the location on the screen and detects collision

wave movement

< toggling the follow me button

< starting the follow timer

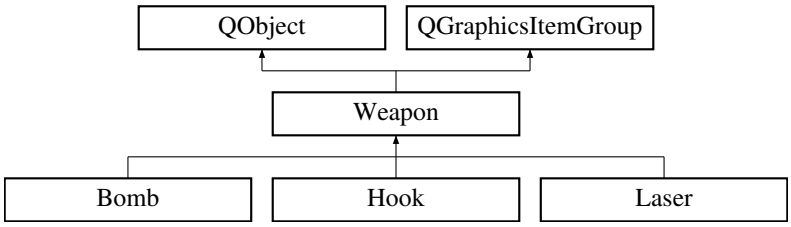
< removing the virus from the scene

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

- virus.h
- virus.cpp

4.30 Weapon Class Reference

Inheritance diagram for Weapon:



Public Slots

- virtual void **update** ()

Public Member Functions

- **Weapon** (QObject *parent=nullptr)

Public Attributes

- QGraphicsPixmapItem * **head**
- QGraphicsLineItem * **rope**
- QGraphicsItem * **grabbedItem**
- QTimer * **throwTimer**
- QTimer * **prepareTimer**
- QString **name**
- bool **thrown**

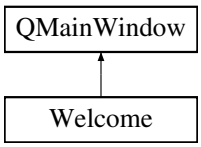
- bool **grabbingItem**
- bool **ready**
- int **step**
- int **strength**

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

- weapon.h
- weapon.cpp

4.31 Welcome Class Reference

Inheritance diagram for Welcome:



Public Member Functions

- **Welcome** (QWidget ∗parent=0)

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

- welcome.h
- welcome.cpp

Chapter 5

File Documentation

5.1 babyspongebob.cpp File Reference

BabySpongeBob class definition.

```
#include "babyspongebob.h"
#include "grabbable.h"
#include <QGraphicsScene>
```

5.1.1 Detailed Description

BabySpongeBob class definition. a BabySpongeBob is an object in the game that resides on the side of the scene, and should be fed healthy items to win the game

Author

Bilal Natafgi

Date

10-4-2018

5.2 bacteria.cpp File Reference

bacteria class definition

```
#include "bacteria.h"
#include "stdlib.h"
#include "game1.h"
#include "game2.h"
#include "hook.h"
```

5.2.1 Detailed Description

bacteria class definition a bacteria is an object in the game that gets eaten by the bear when they collide

Author

Abdel Jawad Alami

Date

22-2-2018

5.3 bomb.cpp File Reference

Bomb class definition.

```
#include "bomb.h"
#include <QPen>
#include "spongeBob.h"
#include <game2scene.h>
#include "huItem.h"
#include <QPointer>
#include "grabbable.h"
```

5.3.1 Detailed Description

Bomb class definition. a Bomb is an object in the game that can be shot if a specific score is reached. it cleares the area it hits

Author

Bilal Natafgi

Date

12-4-2018

5.4 cheat.cpp File Reference

cheat class definition

```
#include "cheat.h"
#include "ui_cheat.h"
```

5.4.1 Detailed Description

cheat class definition TODO

Author

Bilal Natafgi

Date

22-3-2018

5.5 description.cpp File Reference

Description class definition.

```
#include "description.h"
#include "ui_description.h"
#include "game1.h"
```

5.5.1 Detailed Description

Description class definition.

Author

Bilal Natafgi

Date

21-3-2018

5.6 error.cpp File Reference

Error class definition.

```
#include "error.h"
#include "ui_error.h"
```

5.6.1 Detailed Description

Error class definition.

Author

Bilal Natafgi

Date

21-3-2018

5.7 fungus.cpp File Reference

bacteria class definition

```
#include "fungus.h"
#include "stdlib.h"
#include "game1.h"
#include "game2.h"
```

5.7.1 Detailed Description

bacteria class definition a fungus is an object in the game that if eaten by the spongebob, spongebob would loose a life, total strength, and aquariom cleanlines would decrease by half

Author

Abdel Jawad Alami

Date

25-3-2018

5.8 game1.cpp File Reference

game1 class definition

```
#include "game1.h"
#include "ui_game1.h"
#include <QGraphicsItem>
#include <QGraphicsRectItem>
#include <QGraphicsScene>
#include <QGraphicsView>
#include "game1scene.h"
#include "gameview.h"
```

5.8.1 Detailed Description

game1 class definition the class game1 is used to select the difficulty of the game, add the score and create the scene of game1

Author

Bilal Natafgi

Date

21-2-2018

5.9 game1scene.cpp File Reference

game1scene class definition

```
#include "game1scene.h"
#include "stdlib.h"
#include <QGraphicsProxyWidget>
#include <QGraphicsEllipseItem>
#include <QPainter>
#include <QGraphicsLinearLayout>
#include <QColorDialog>
#include "game1.h"
#include "scores.h"
```

5.9.1 Detailed Description

game1scene class definition a game1scene contains all the interactions in the game

Author

Abdel Jawad Alami

Date

22-2-2018

5.10 game2.cpp File Reference

Game2 class definition.

```
#include "game2.h"
#include "ui_game2.h"
#include "gameview.h"
#include "game2scene.h"
```

5.10.1 Detailed Description

Game2 class definition. a Game2 is where the user selects the strength of the game to play

Author

Bilal Natafgi

Date

22-2-2018

5.11 gamemenu.cpp File Reference

Gamemenu class definition.

```
#include "gamemenu.h"
#include "ui_gamemenu.h"
#include "game1scene.h"
#include "game2scene.h"
#include "gameview.h"
#include "game.h"
#include "error.h"
```

5.11.1 Detailed Description

Gamemenu class definition. a Gamemenu is where the user can enter the games, or check his highscore and game descriptiond

Author

Bilal Natafgi

Date

20-2-2018

5.12 gameview.cpp File Reference

gameview class definition

```
#include "gameview.h"
#include "ui_gameview.h"
```

5.12.1 Detailed Description

gameview class definition a gameview is where the game scene resides

Author

Bilal Natafgi

Date

20-2-2018

5.13 grabbable.cpp File Reference

Grabbable class definition.

```
#include "grabbable.h"
#include "huItem.h"
#include "bomb.h"
#include "laser.h"
```

5.13.1 Detailed Description

Grabbable class definition. the grabbable class is excited when an item collides with a weapon or a hook

Author

Bilal Natafgi

Date

20-2-2018

5.14 header.cpp File Reference

Header class definition.

```
#include "header.h"
#include "user.h"
#include "game1scene.h"
#include "game2scene.h"
#include <QApplication>
#include "game1.h"
#include "game2.h"
```

5.14.1 Detailed Description

Header class definition. the Header contains most of the metrics in the game. it contains the difficulty of the game, the username of the player, the timer, the cleanliness of the tank in game one, and henumber of healthy items fed to the baby in game 2. the header contains all items that are located at the top of the game scene.

Author

Bilal Natafgi

Date

20-2-2018

5.15 history.cpp File Reference

history class definition

```
#include "history.h"
#include "ui_history.h"
```

5.15.1 Detailed Description

history class definition the history contains previous scores of a given player.

Author

Bilal Natafgi

Date

23-2-2018

5.16 home.cpp File Reference

home class definition

```
#include "home.h"
#include "ui_home.h"
```

5.16.1 Detailed Description

home class definition the home page is where the user is redirected after signing in.

Author

Bilal Natafgi

Date

23-2-2018

5.17 hook.cpp File Reference

hook class definition

```
#include "hook.h"
#include <QPen>
#include "spongeBob.h"
#include <QGraphicsScene>
#include "bacteria.h"
#include "huItem.h"
#include "virus.h"
#include "fungus.h"
#include <QPointer>
#include "grabbable.h"
```

5.17.1 Detailed Description

hook class definition the hookis a type of weapons by which the player grabs items

Author

Bilal Natafgi

Date

13-4-2018

5.18 huItem.cpp File Reference

huItem class definition

```
#include "huItem.h"
#include "stdlib.h"
#include "game1.h"
#include "game2.h"
```

5.18.1 Detailed Description

huItem class definition a huItem is an object in the game that gets eaten by the spongebob when they collide, or by baby spongebob

Author

Abdel Jawad Alami

Date

22-2-2018

5.19 scores.cpp File Reference

scores class definition

```
#include "scores.h"
```

5.19.1 Detailed Description

scores class definition the scores class

Author

Bilal natafgi

Date

26-2-2018

5.20 signin.cpp File Reference

SignIn class definition.

```
#include "signin.h"
#include "ui_signin.h"
#include "error.h"
```

5.20.1 Detailed Description

SignIn class definition. the SignIn class

Author

Bilal natafgi

Date

20-2-2018

5.21 signup.cpp File Reference

signup class definition

```
#include "signup.h"
#include "ui_signup.h"
```

5.21.1 Detailed Description

signup class definition the signup class

Author

Bilal natafgi

Date

20-2-2018

5.22 spongeBob.cpp File Reference

spongeBob class definition

```
#include "spongeBob.h"
#include "game1scene.h"
#include "game1.h"
#include "game2.h"
#include "laser.h"
#include "hook.h"
#include "bomb.h"
#include "game2scene.h"
```

Variables

- QSet< int > pressedKeys

  spongeBob::keyPressEvent

5.22.1 Detailed Description

spongeBob class definition a spongeBob instance

Author

Abdel Jawad Alami

Date

18-3-2018

5.22.2 Variable Documentation

5.22.2.1 QSet<int> pressedKeys

spongeBob::keyPressEvent

Parameters

event	the keypress event detects all key strokes on the keyboard. we are only interested in the up,down,left, and right strokes when one of those keys is pressed, the bear moves accoringly when possible if the location of the bear is on the edges of the screen, the position isnt updated if this will result in leaving the screen

5.23 virus.cpp File Reference

virus class definition

```
#include "virus.h"
#include "stdlib.h"
#include "game1.h"
#include "game2.h"
```

5.23.1 Detailed Description

virus class definition a virus is an object in the game that if eaten by spongebob, makes the bacteria follow him.

Author

Abdel Jawad Alami

Date

18-3-2018

5.24 weapon.cpp File Reference

weapon class definition

```
#include "weapon.h"
```

5.24.1 Detailed Description

weapon class definition this weapon class is an abstract class that all weapons inherit from.

Author

Abdel Jawad Alami

Date

18-3-2018

5.25 welcome.cpp File Reference

welcome class definition

```
#include "welcome.h"
#include "ui_welcome.h"
#include "signup.h"
#include "signin.h"
```

5.25.1 Detailed Description

welcome class definition welcome class with its functions

Author

Bilal Natafgi

Date

18-3-2018