Source code documentation

Game class:

The Game class inherits QGraphicsView class. It creates a scene to the view. The constructor creates buttons and text labels to the scene.
addItems(QVector<QVector<tile>> map)
The addItems function gets a two-dimensional vector of tile-objects as a parameter. The function adds those tiles on the scene.

readWaves(const QString filename, QVector<QVector<int>> path)
The readWaves function gets a const QString string and 2-dimensional vector of integers as parameters. The vector represents the path that has been created and the filename is the name of the file corresponding to the difficulty level the player has chosen. The function reads the enemy waves from the file and adds them to a waves_ 2-dimensional vector.

Handle_screen_click(double x, double y)
Handle_screen_click function gets two double values as parameters. X and Y values represent the point where the screen is clicked. The function gets the QGraphicsItem that is at the point with the itemAt method. If there is no QGraphicsItem where the screen is clicked, the function does nothing. If a TileGraphicsItem is clicked and also one of the tower buttons is clicked, the function adds tower to the tile. If the tile is a path tile, the tower is not added. If a tower is clicked, then its range is shown. If a range of a tower is clicked, it will be hidden. However if a tower is clicked and also a sell button has been clicked, the tower will be sold. Buying and selling a tower will reduce or increase the amount of money.

StartGame(QString filename, int diff, int maplevel)
The StartGame function is always called in the mainwindow, when a player clicks the start game button. The parameters are a QString filename and two integer numbers corresponding to the map and difficulty levels the player has chosen in the mainwindow. With the maplevel parameter it is possible to choose the correct file to create the map and the diff value is used when creating a filename to call readWaves function. Startgame function creates a map with the correct filename and then calls readWaves.

createEnemies()
The createEnemies function is always called when the player clicks the *start next wave* button. The enemiesOnPath vector has to be empty (the earlier wave has ended) for the function to work properly. The function takes the first element of the two-dimensional vector and deletes it from the waves_ list. However, it adds the one-dimensional list to enemies parameter. Then the function creates a timer and calls the spawn() function every 500 ms.

spawn()
The spawn function is always called from the createEnemies function. It takes the first enemy letter from the enemies list, creates a corresponding enemy, adds it to the screen

and deletes the letter from the enemies list. If the enemies list is empty, the function disconnects the timer, and it is not called anymore.

killEnemy(Enemy *enemy)
The killEnemy function deletes the enemy if it is shot by a tower. It deletes it from the enemiesOnPath_ list and adds the amount of money the enemy was worth to the player's money. The function deletes the enemy. It also calls winningmessage() function if the enemy was the last one in the last wave.

Tower class:

The tower class inherits QObject and QGraphicsPixmapItem classes. The tower constructor gets a filename as a parameter and creates a pixmapitem from the filenames file. Different towers are created from different files. The constructor sets the scale of the qpixmapitem and also saves given parameters to private variables. The tower is connected to the attack_target function.

attack_target(QVector<Enemy*> enemiesOnPath)
Calls acquire_target() function.

acquire_target()
The acquire_target function gets a list of QGraphicsItems that are on its range with collidingItems method. The function searches the closest enemy on its range and if there is one, it shoots towards the enemy. The attack_point_ is set to the point where the closest enemy is and then a fire() function is called. After this the target enemys takeDamage() is called. This function is called 1000ms*fire_rate_, so different towers shoot with different time intervals.

fire()
The fire function is called in acquire_target() function when the tower has a target. The fire function creates a bullet and sets its position in the middle of the tower. Then the bullet item is rotated towards the target enemy with the bullet's setRotation function with the correct angle. Then the bullet is added to the game scene.

Basic class:

A subclass of a Tower class. Inherits all tower functions. Basic class represents a basic tower that can make damage worth 1 hitpoint and has normal sized range. Its fire rate is 1. The price of a basic tower is 400.

Quicky class:

A subclass of a Tower class. Inherits all tower functions.Quicky tower can make damage worth 1 hitpoint but has fire rate 0.5 (fires 2 times faster than basic tower). The price of a quicky tower is 1300.

Sniper class:

A subclass of a Tower class. Inherits all tower functions. Sniper can make damage worth 3 hitpoints. It also has a bigger range than other towers. Its fire rate is 1.5. Sniper's ap value is true, which means it can make damage to green enemy.

Piercer class:

A subclass of a Tower class. Inherits all tower functions. Piercer can make damage worth 2 hit points and has a 1.2 fire rate. Its range is the same size as basic's and quicky's. Piercer's ap value is true, which means it can make damage to green enemy.

Enemy class:

The Enemy class inherits the QObject and QGraphicsPixmapItem classes. Depending on the type of the enemy, a square qpixmapitem filled with the correct colour is created. Every enemy gets the map path as a parameter and saves it to a path_ private variable. The enemy item is connected to the Move() function in the constructor.

QPointF getNextPoint()
This function gets the next QPointF point from a path_ list and deletes it from the list. The function returns the point.

void takeDamage(int dmg)
The takeDamage function is called form tower acquire_target() function. The tower is being shot and it takes damage. Different enemies have different amounts of hp_ and also different towers can shoot different amounts of damage. The integer dmg the function gets as a parameter tells how much damage the tower is able to do. The amount of damage will be reduced from the enemy's hp_. If the hp_ number is zero or less, the enemy will be killed and the game's killEnemy() function is called.

void Move();
The move function moves the enemy towards its next destination. If the enemy is close enough to the destination, the function will give a new destination for it. The enemy is rotated towards the next destination. The enemy is moved closer to the destination. However, if the current destination is the last point of the path, the enemy will be deleted from the scene and enemiesOnPath list and the reduceLives function is called.

Red class:

Subclass of an Enemy class. Red is a basic enemy that has 1 hit point and dies when any tower shoots it once.

Blue class:

Subclass of an Enemy class. Blue is a basic enemy that has 2 hit points and two damage points are needed to kill the enemy.

Green class:

Subclass of an Enemy class. Blue is a basic enemy that has 3 hit points. The green enemy has armour which means that only sniper and piercer towers can make damage to the green enemy.

White class:
Subclass of an Enemy class. Blue is a basic enemy that has 4 hit points and four damage points are needed to kill the enemy.

Boss class:
Subclass of an Enemy class. Blue is a basic enemy that has 10 hit points and ten damage points are needed to kill the enemy.

Bullet class:

The bullet class inherits QObject and QGraphicsPixmapItem classes. A picture of an arrow is set as the pixmap and a timer is created to call the move function.

void move()
The move function moves the bullet to the same direction it is rotated. The bullet is rotated in the tower class when it is created. The function uses the rotation() method to get the angle of rotation. The function is called every 10ms.

TileGraphicsItem class:

TileGraphicsItem class inherits QGraphicsRectItem class. This class is a visual representation of a tile class. Each tile object has a corresponding TileGraphicsItem object. TileGraphicsItem object is couloured blue if it is not a path, and grey if it is a path.

Map class:

Map class creates a map object from a file.

int readMap(const QString filename):
readMap file gets a QString filename as a parameter. The parameter tells the name of the file that is used to create the map. ReadMap function searches the start point of the path from the given file and also looks for the rest of the path. A path is ended to a ending tile. The path is saved to a path_ two-dimensional vector.

Tile class:

The tile class represents one tile in the map. A tile can be path or not path and it can be reserved or not reserved. If a tower is placed on a tile, then it is reserved. Each tile object also has x and y coordinates and size. The coordinates determine where the tile is placed on the map. All tiles are the same size.