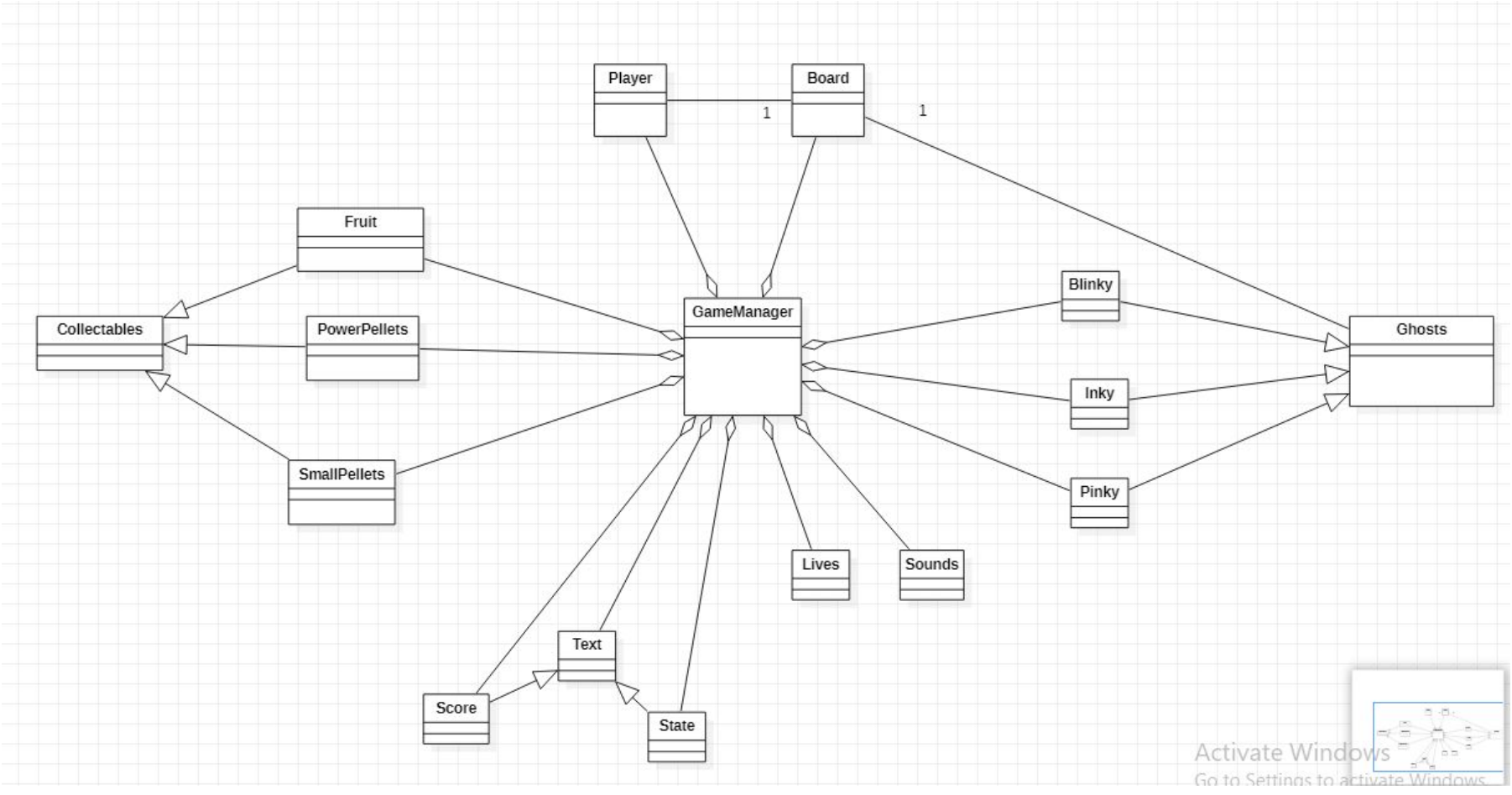
Pacman Project

CS 110

Developed by:
Abdelrahman Abdelmonem
Abdallah Taha
Amer Elsheikh

Supervised by:
Dr. Howaida Ismail

General Structure



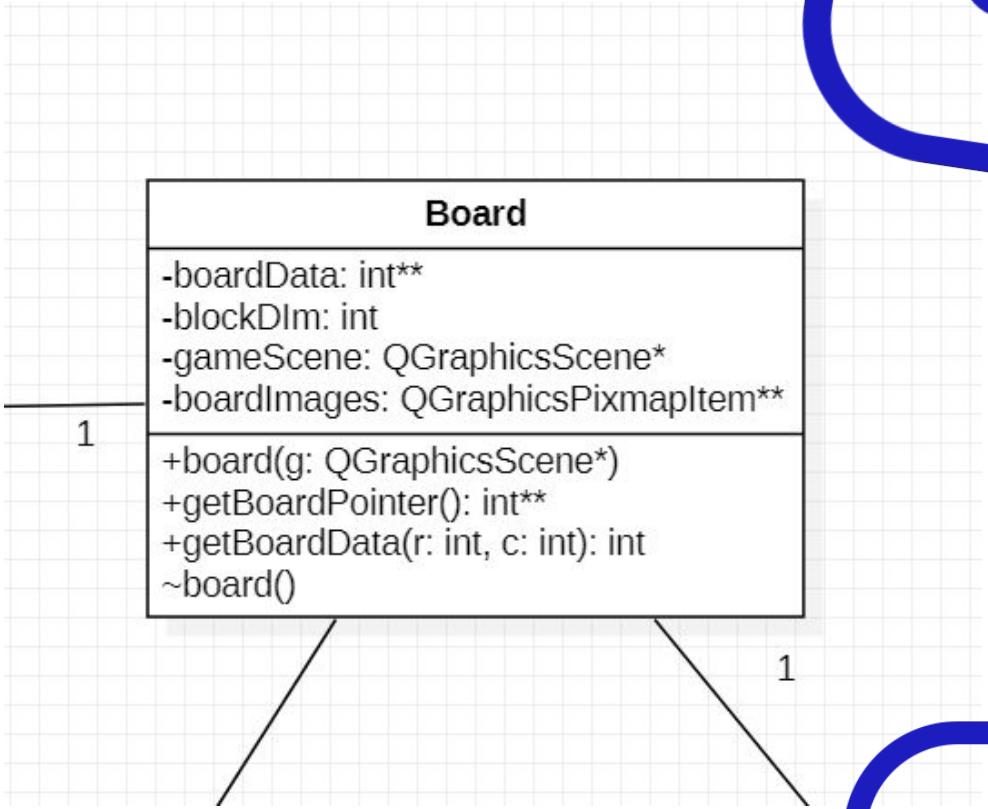
Board

-Reads and saves board.txt using QFile and QTextStream.

-Initializes QPixmaps with the right size and image.

-Sets them in QGraphicsPixmapItem array with their correct position and adds them to the scene.

getBoardData and getBoardPointer returns boardData.



Player

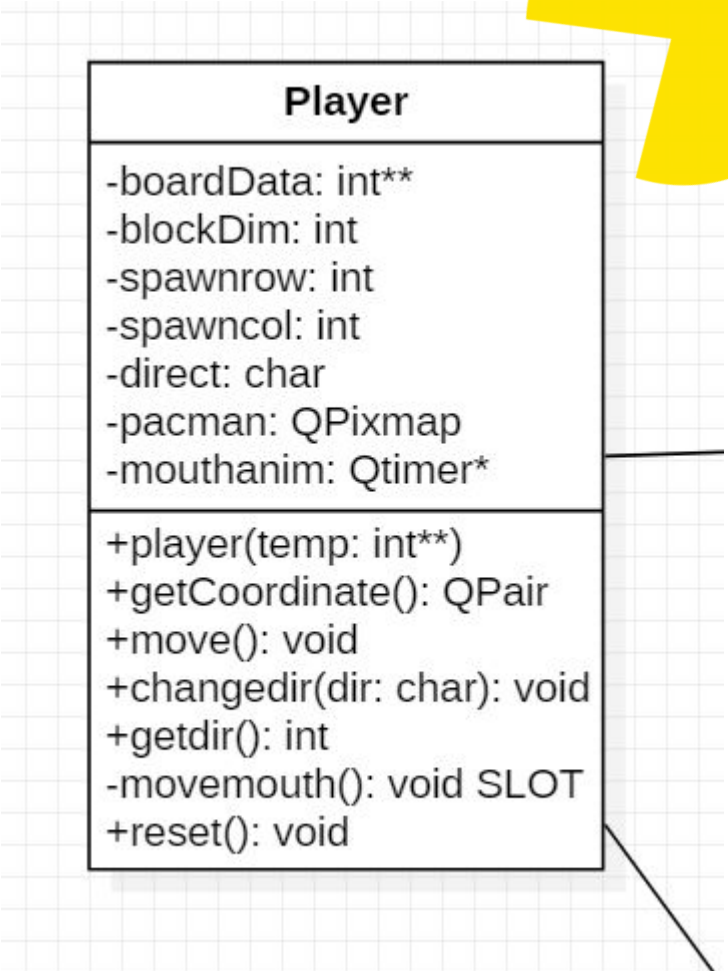
Inherits `QGraphicsPixmapItem`.

When player is initialized it takes the board data as an argument and stores it.

Pacman image is loaded and pacman is spawned in its default spawn.

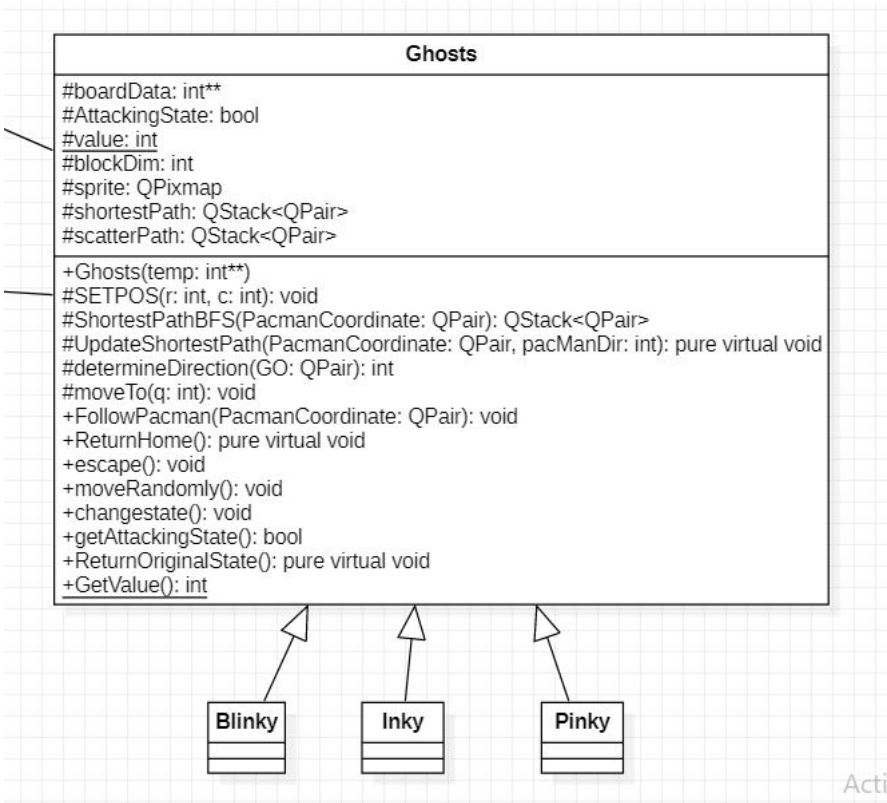
We can get its direction and coordinates using `getdir()` and `getCoordinate()`.

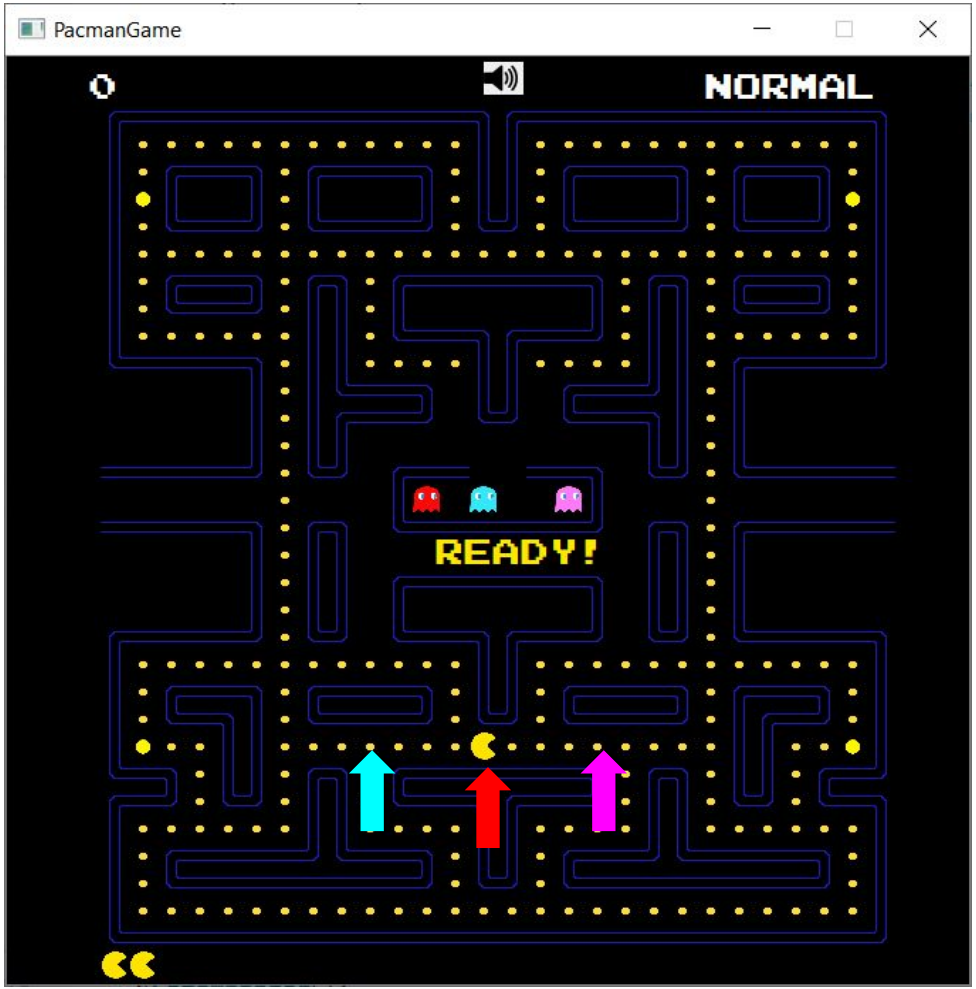
`movemouth()` handles pacman smooth motion.



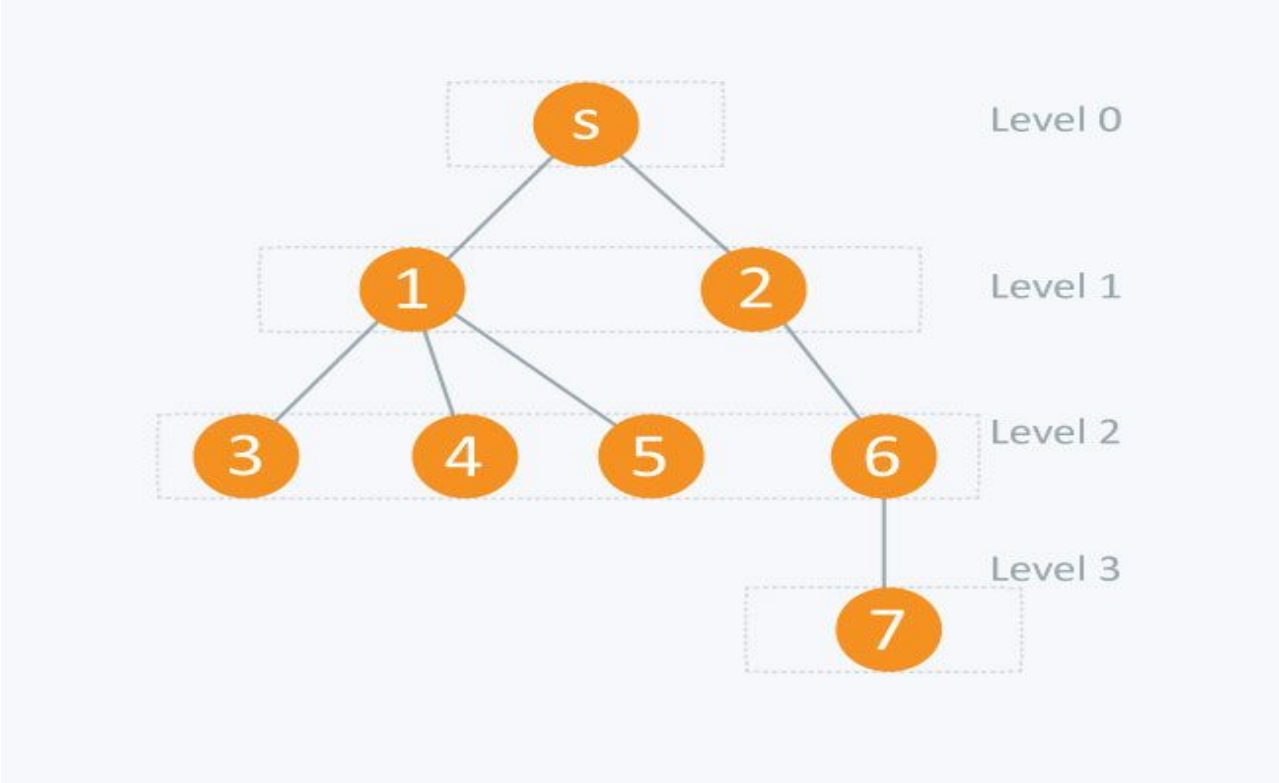
Ghosts

- Inherits from QGraphicsPixmapItem
- Value: static and doubles
- Smoothness
- Escaping mode:
  - changestate()
  - escape(): away then randomly
  - Not eaten: ReturnOriginalState():pure
  - Eaten: ReturnHome(): pure
- Following mode:
  - ScatterPath
  - FollowPacman()
  - UpdateShortestPath(): pure
  - ShortestPathBFS()



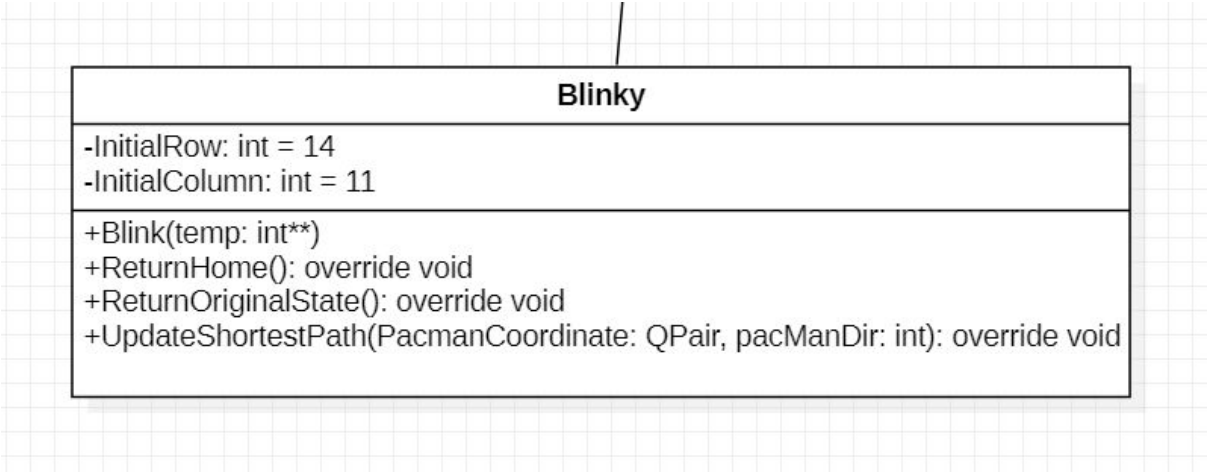

PacmanGame
0
NORMAL
READY!

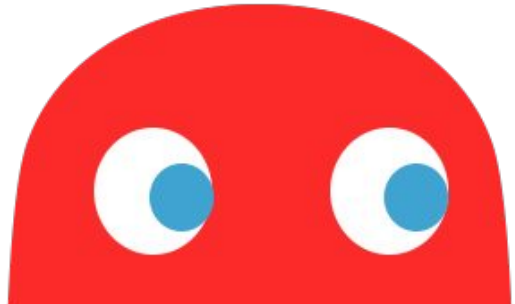
ShortestPathBFS



The graph is unweighted and connected; the nearest node is always at the front of the queue

Blinky


Blinky
-InitialRow: int = 14
-InitialColumn: int = 11
+Blink(temp: int**)
+ReturnHome(): override void
+ReturnOriginalState(): override void
+UpdateShortestPath(PacmanCoordinate: QPair, pacManDir: int): override void

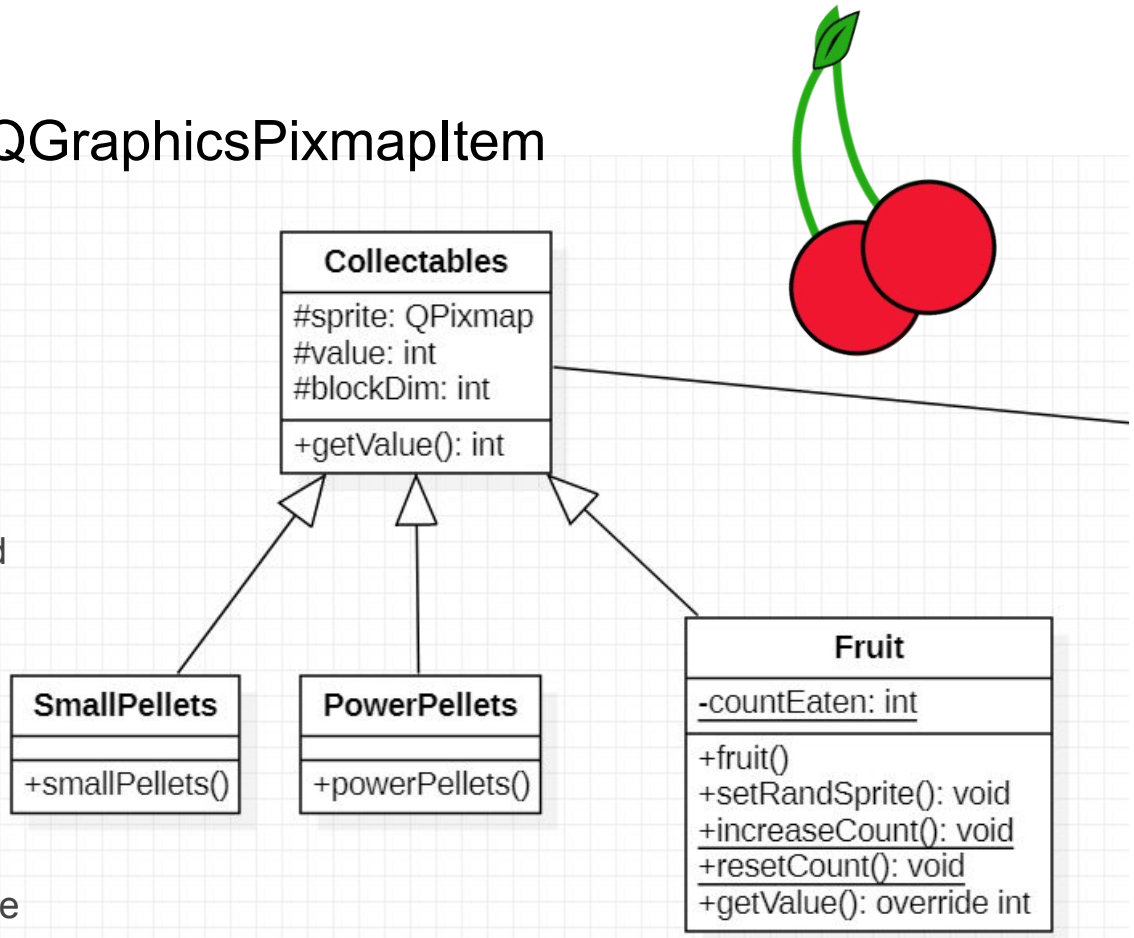


Collectables - inherits QGraphicsPixmapItem

Pellets:
-Initilizes value and QPixmap with right image and the right size.
-Sets QPixmap to QGraphicsPixmapItem.

Fruit:
-one fruit object that changes position and image randomly.
-setRandSprite, generates a random number by qrand() and loads a random image with the right size.
-Keep track of eaten fruit.

*Positions and adding to the scene is done in gamemanager..



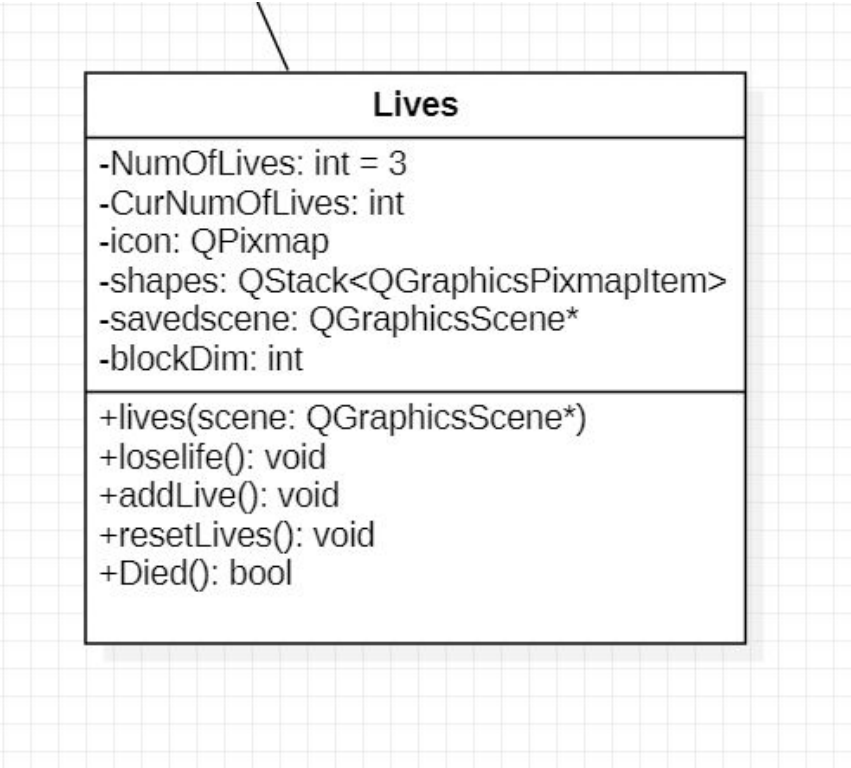
Lives

Lives is a simple class.

It is created to keep track of lives.

Lives class takes the current scene as an argument for the constructor.

Lives are add to the scene based on a stack.



Text

PACMAN

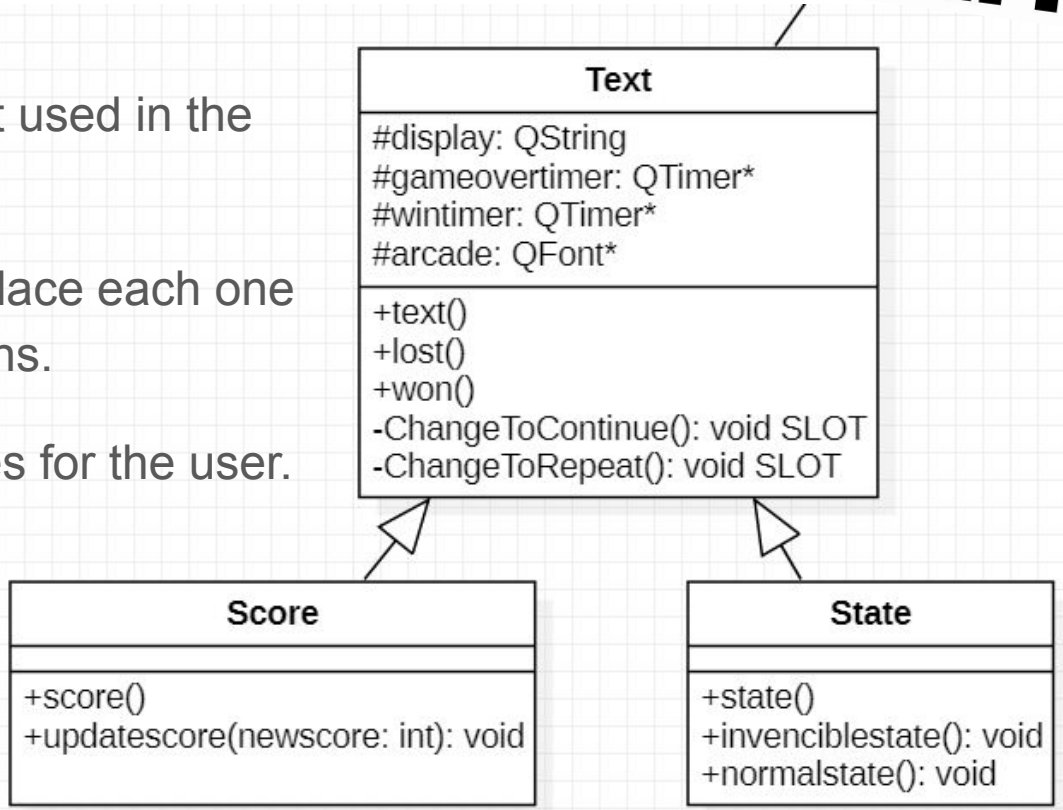
Text has the format of any text used in the game.

Score and State differ in the place each one appears on and their operations.

Text is used to show messages for the user.

Score shows player's score.

State shows pacman's state.

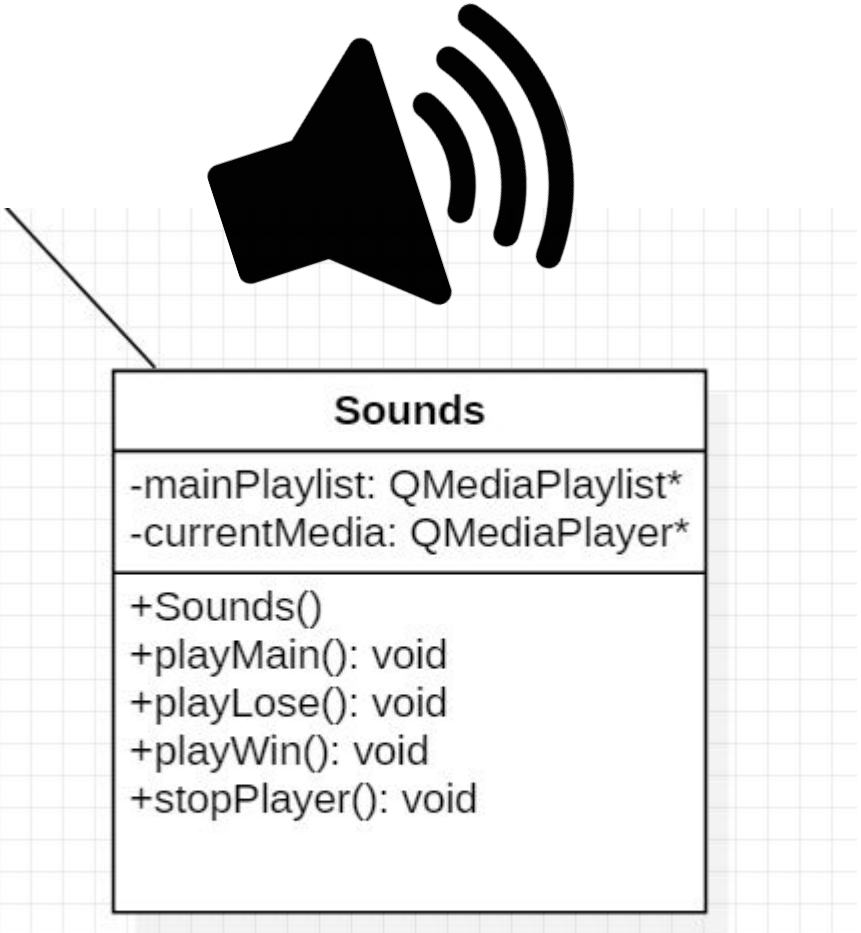


Sound

The sound class is pretty simple and fun.

-Initializes QMediaPlaylist (mainPlaylist) and adds to it our sounds/music.
-Creates a QMediaPlayer (currentMedia) and connects it to our playlist.

Functions - They do pretty much the same thing, they set a specific music and playBackMode in the QMediaPlaylist and then calls currentMedia.play



GameManager

Functions

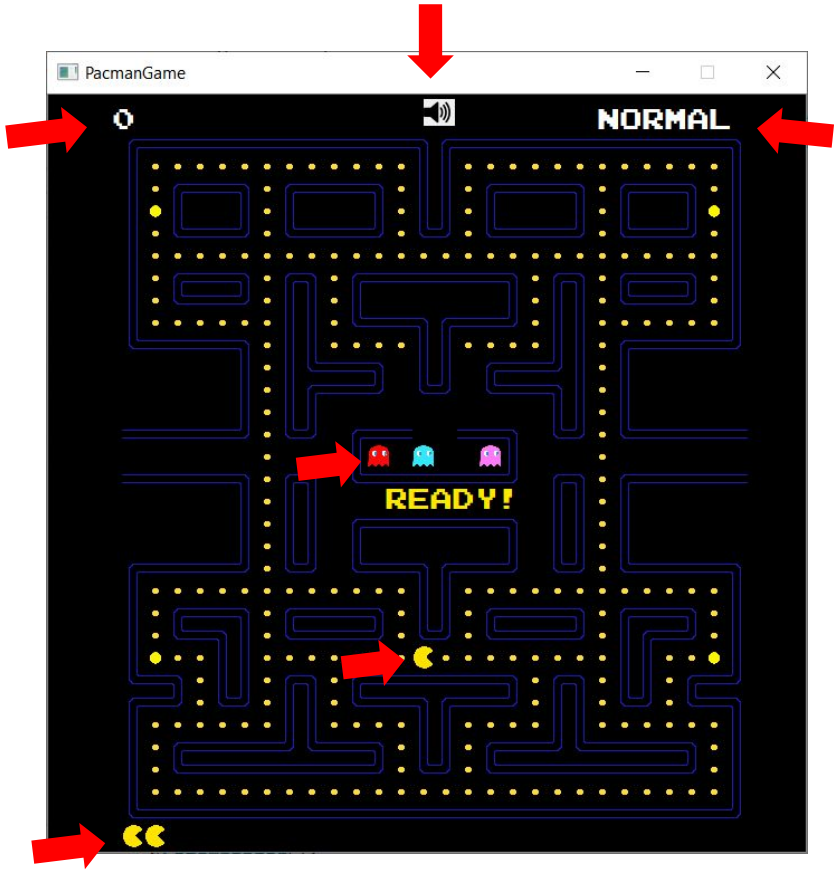
GameManager()

Sets puts the scene on the view and sets its color to black.

Creates instances from the board, ghosts, pacman, lives, texts, and the mute button.

Initializes and connects timers to their slots.

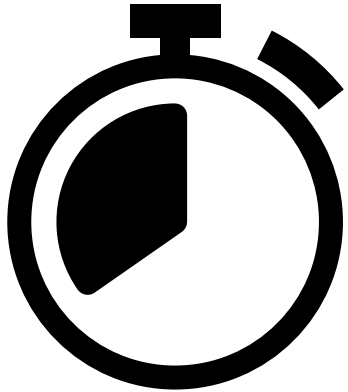
Now, the game is ready!



Timers

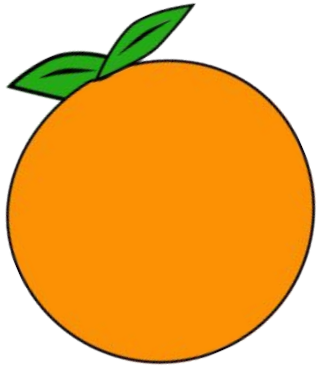
`timer`

`advance()`



advance()

- This is our main function, and it is called randomly every 18 milliseconds.
- It first checks whether the user exceeded a multiple of 10 thousand or not, and assign new lives accordingly
- Then it checks three types of collisions and handles them:
  - Small Pellets
  - Power Pellets
  - Ghosts
  - fruits
- Then it checks if the user has died and act accordingly
- If not died, then it checks if he won (eaten all pellets) and act accordingly
- If not both, then it calls the functions that move the pacman and the ghosts according to their state (attacking or escaping)



Collision with small pellets and fruit

```
if( typeid(*collidedItems[i]) == typeid(smallPellets)){

    scene->removeItem(collidedItems[i]);
    playerScore += smallPelletsarr[0].getValue();
    UneatenPellets--;
```

```
}else if(typeid(*collidedItems[i]) == typeid(fruit)){    ⚠ expression with side
        scene->removeItem(collidedItems[i]);
        fruit::increaseCount();//increase the count of fruit already appeared
        playerScore += fruitInstance.getValue();
        int tempT = (qrand()%5) + 14;
        timerFruit->start(1000*tempT);//timer works randomly
}
```

Collision with power pellets

```
}else if(typeid(*collidedItems[i]) == typeid(powerPellets)){

    scene->removeItem(collidedItems[i]);
    UneatenPellets--;
    Ghosts::SetValue();//resets value to 200
    pacstate->invenciblestate();
    playerScore += powerPelletsarr[0].getValue();
    InkyInstant->changestate();
    InkyInstant->escape();
    PinkyInstant->changestate();
    PinkyInstant->escape();
    BlinkyInstant->changestate();
    BlinkyInstant->escape();
    timerGhostState->start(9000);// they remain escaping for 9 seconds
    //or until got eaten, whicever nearer ;)
```

Collision with Ghosts (Blinky)

```
}else if(typeid(*collidedItems[i]) == typeid(Blinky)){

    if(BlinkyInstant->getAttackingState() == 0){
        BlinkyInstant->ReturnHome();
        playerScore += Ghosts::GetValue();
         Ghosts::DoubleValue();
    }else{
        remlives->loselife();
        resetGame(0);
    }
```

resetGame(bool win)

Called when pacman is eaten or when there is no more pellets.

It returns everything to its place.

If the player didn’t win we delay its start for smoother gameplay.

If pacman lost

```
if(remlives->Died()){
    if(MusicWorks){
        musicManager->playLose();//play the losing music
    }
    gamestate->lost();//show gameover; after 2 seconds, Again Y/N
    scene->addItem(gamestate);
    timer->stop();
    timerFruit->stop();
    pacman->endanim();
    scene->removeItem(&fruitInstance);
```

If pacman won

```
}else if(UneatenPellets==0){
    if(MusicWorks){
        musicManager->playWin();
    }
    gamestate->won();
    scene->addItem(gamestate);
    timer->stop();
    timerFruit->stop();
    pacman->endanim();
    resetGame(1);
    scene->removeItem(&fruitInstance);
}
```

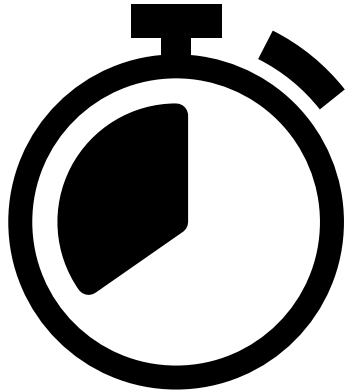
Collision with Ghosts (Blinky)

```
}else if(typeid(*collidedItems[i]) == typeid(Blinky)){

    if(BlinkyInstant->getAttackingState() == 0){
        BlinkyInstant->ReturnHome();
        playerScore += Ghosts::GetValue();
         Ghosts::DoubleValue();
    }else{
        remlives->loselife();
        resetGame(0);
    }
```

If pacman hasn’t won or lost yet

```
else{

    if(InkyInstant->getAttackingState())
        InkyInstant->FollowPaceman(pacman->getCoordinate(), pacman->getdir());
    else
        InkyInstant->escape();

    if(PinkyInstant->getAttackingState())
        PinkyInstant->FollowPaceman(pacman->getCoordinate(), pacman->getdir());
    else
         PinkyInstant->escape();

    if(BlinkyInstant->getAttackingState())
        BlinkyInstant->FollowPaceman(pacman->getCoordinate());
    else
        BlinkyInstant->escape();

    pacman->move();// move pacman
}
```

Timers



`timer` → `advance()`

`timeFruit` → `createFruit()`

createFruit()

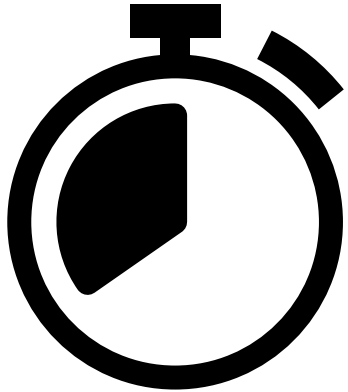
This function is called randomly every 14 to 18 seconds.

Calls setRandSprite on our fruit instance, and then sets a random position on the board and adds it to the scene.

It resets the timer with a different value between 14 and 18 sec.

If the fruit already exists on the board it will change its position and its image randomly.

Timers



`timer` → `advance()`

`timeFruit` → `createFruit()`

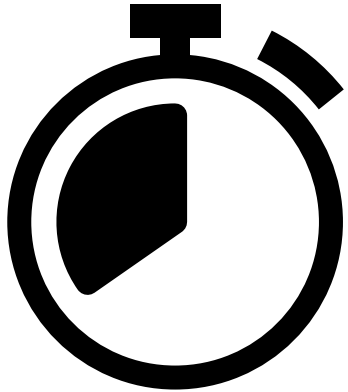
`timerGhostState` → `ghostStateTimeout()`

ghostStateTimeout()

If there is a ghost that is still scared, its state is changed back to chasing pacman.

Changes the text on the screen.

Timers



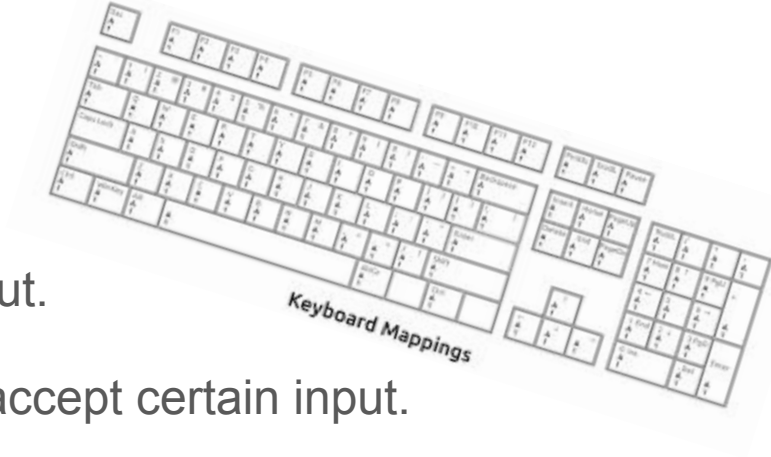
`timer`	→	`advance()`
`timeFruit`	→	`createFruit()`
`timerGhostState`	→	`ghostStateTimeout()`
`delay`	→	`delayStart()`

delayStart()

Designed to delay the start of the game after reset to give the better playing experience.

Starts pacman’s mouth animation and the game’s main timer again.

KeyPressEvent()



It is a vital and crucial function as it handles all input.

There are 4 different (phases), in each phase we accept certain input.

These phases are:
-The game hasn’t started: we check for space key.
-The game is running: we check directional inputs and change pacman direction.
-The player is dead: we check Y/N key and either start a new game and reset everything (score, lives, fruit) or close it.
-The player has won: we check Y/N key and either continue the game or close it.

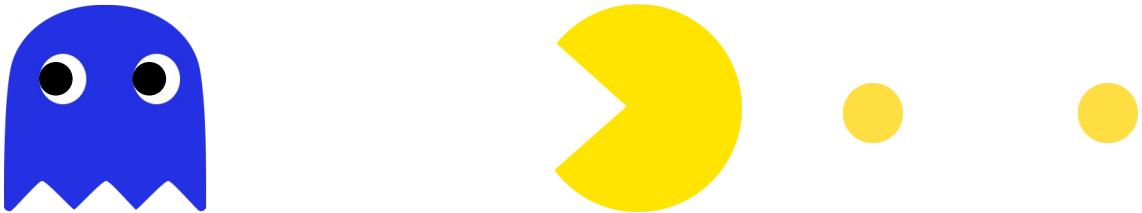
*player has won equivalent uneaten pellets = 0;

startAgain()

Called if the user wants to play again.

It fills the pellets again and starts the timers.

It plays the appropriate music as well.



FillPellets()

When it is called it adds and sets the position of the pellets in the scene.

We initialize the number of unEatenPellets = 248.

We loop over the boardData.

We check if the row and col has the right value and set position if it does.

OnMusicClicked()

This is a slot function that is connected to the signal clicked() of a QPushButton.

It mutes or plays the sounds of the game once the button is clicked.

The QPushButton is initialized in the gameManager constructor.

When it is called it checks bool MusicWorks. In case it is true, it changes the icon, sets MusicWorks to false, and calls stop on sound class.