QTetris

v2.0

Généré par Doxygen 1.8.3

Lundi Mars 18 2013 21 :56 :41

Table des matières

Chapitre 1

Projet IN204 : QTetris

Mini moteur de jeu & Demonstrateur du Tetris.

Auteur

Mengmeng Zhang & Marouane Fazouane

Version

2.0

Date

18 March 2013

1.1 Introduction

Le programme est un demonstrateur du projet Tetris.

1.2 Details

- Projet : Tetris
- Langage : C++11/ Qt
- IDE : QtCreator
- Source Control : GIThttps ://github.com/
- Documentation : Doxygen
- ModéliseurUML : Visio & Umbrello

1.2.1 Roadmap

- Installer les bibliothèques sur nos ordinateurs
- Prendre un compte Micro sur github et configurer l'IDE
- Se mettre d'accord sur le CodingStyle
- Compléter la description des modules
- Compléter l'UseCase
- Commencer les diagrammes de classes
- Réaliser un mini Tetris (très simple jeu) en mode console : Premier prototype du jeu
- But :
  - tester la suppression de lignes ;
  - tester les actions (rotations, déplacements. . .)

  - tester la chute
  - tester la fin du jeu
- Description :
  - Carte : matrice
  - Pièces : des positions : x,y (Model) ; des étoiles (View)
  - Commandes données sur la console
- Réaliser un prototype du système de menus∗
- Approuver les diagrammes et implémenter le diagramme de classe en C++
- Commencer la documentation
- Travailler la GUI basique et l'intégrer au code
- Développer la GUI et le système de menu
- Rajouter la configuration par xml et un gestionnaire de son

Chapitre 2

Index hiérarchique

2.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 3

Index des classes

3.1 Liste des classes

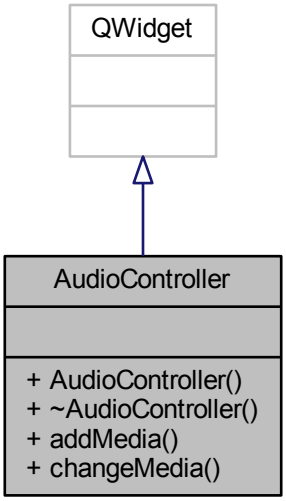
Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 4

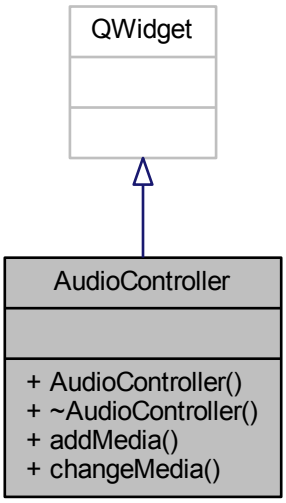
Documentation des classes

4.1 Référence de la classe AudioController

Graphe d'héritage de AudioController :



Graphe de collaboration de AudioController :



Connecteurs publics

– void **addMedia** (const QString &m)
– void **changeMedia** (const QString &m)

Signaux

– void **mediaChanged** (const QString &m)

Fonctions membres publiques

– **AudioController** (QWidget ∗parent=0)

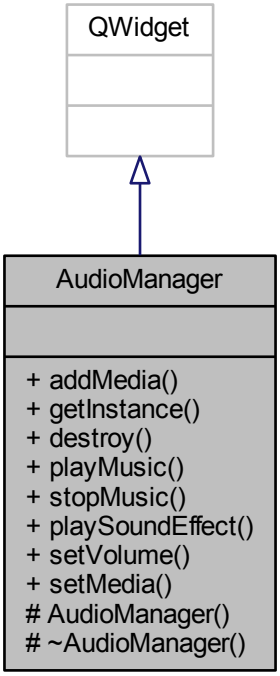
4.1.1 Description détaillée

Définition à la ligne 9 du fichier AudioController.h.

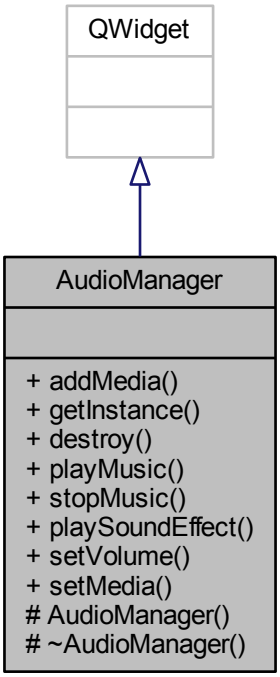
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ExtraWidgets/AudioController.h
– Source/ExtraWidgets/AudioController.cpp

4.2 Référence de la classe AudioManager

Graphe d'héritage de AudioManager :



Graphe de collaboration de AudioManager :



Connecteurs publics

- void **playMusic** ()
- void **stopMusic** ()
- void **playSoundEffect** ()
- void **setVolume** (int v)
- void **setMedia** (const QString &title)

Signaux

- void **mediaChanged** (const QString &title)

Fonctions membres publiques

- AudioManager & **addMedia** (const QString &title, const QString &name)

Fonctions membres publiques statiques

- static AudioManager & getInstance ()
- static void **destroy** ()

Fonctions membres protégées

- AudioManager (QWidget *parent=0)
- virtual ~AudioManager ()

4.2.1 Description détaillée

Définition à la ligne 13 du fichier AudioManager.h.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 AudioManager : :AudioManager (QWidget ∗ *parent* = 0) [protected]

Empty Constructor

Définition à la ligne 7 du fichier AudioManager.cpp.

Voici le graphe des appelants de cette fonction :

4.2.2.2 AudioManager : :∼AudioManager ( ) [protected], [virtual]

Empty Destructor

Définition à la ligne 17 du fichier AudioManager.cpp.

4.2.3 Documentation des fonctions membres

4.2.3.1 static AudioManager& AudioManager : :getInstance ( ) [inline], [static]

Singleton (DesingPattern)

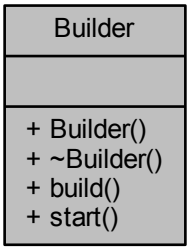
Définition à la ligne 36 du fichier AudioManager.h.

Voici le graphe d'appel pour cette fonction :

La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/AudioManager/AudioManager.h
– Source/AudioManager/AudioManager.cpp

4.3 Référence de la classe Builder

Graphe de collaboration de Builder :



Fonctions membres publiques

– void **build** ()
– void **start** ()

4.3.1 Description détaillée

Définition à la ligne 11 du fichier Builder.h.

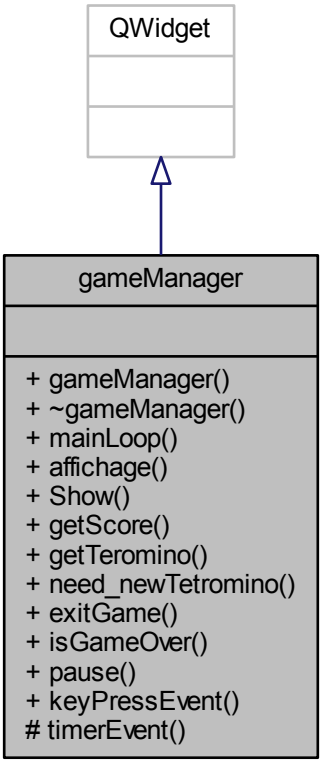
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ResourcesManager/Builder.h
– Source/ResourcesManager/Builder.cpp

4.4 Référence de la classe gameManager

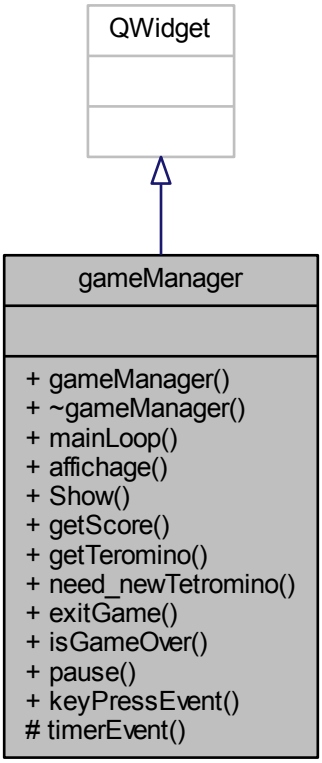
le class gameMangager va gerer les regles du jeu, e.g. le choi de tetromino a l'instant et a l'avenir, la note, la niveau de difficulte, la vitenesse de jeu...

```
#include <gameManager.h>
```

Graphe d'héritage de gameManager :



Graphe de collaboration de gameManager :



Connecteurs publics

– void pause ()
  pause sert a mettre le gamemanager en pause ou a le relancer
– void keyPressEvent (QKeyEvent ∗ev)
  gestion des controles de clavier

Signaux

– void scoreChanged (int)
  scoreChanged va emettre un signal quand on a une nouvelle note
– void nextLevel (int)
  nextLevel va emettre un signal quand on a un nouveau niveau
– void tetrominoChanged (int, int, int)
  tetrominoChanged va emettre un signal quand on a un nouveau tetromino

Fonctions membres publiques

– **gameManager** (QWidget ∗parent=0)
– void mainLoop ()
  mainloop : gerer les fontions ci-dessous 1) verifier si on a besoin un nouveau tetrimino par la valeur de "S.need_new-_tetromino" 2) appeler S.doPhysics() : gerer la physique (gravite+ suppressions de lignes multiples...etc) 3) verifier le nombre de lignes supprime a la meme temps, et calculer la note. 4) appeler isGameOver() : verifier si le jeu est termine
– void affichage ()

affichage Appeler S.show() regulierement pour quelle ait une copie plus recente de son etat lorsqu'on voudra effectivement l'afficher a l'ecran.

– QString Show ()
show : sert d'intermediaire entre la classe Scene et le widget d'affichage
– unsigned int getScore ()
getScore : getter de la valeur courante du score
– Tetromino getTeromino ()
getTeromino getter du tetromino courant
– bool need_newTetromino ()
need_newTetromino
– void exitGame ()
exitGame : juste pour debug Affichage "Game Over" si le jeu est termine
– bool isGameOver ()
IsGameOver Si le valeur de "S.gameOver" = true, il va arreter tous les timeurs et fair "var_gameOver" = ture.

Fonctions membres protégées

– void timerEvent (QTimerEvent ∗event)
timerEvent va gerer les "timer event"

4.4.1 Description détaillée

le class gameMangager va gerer les regles du jeu, e.g. le choi de tetromino a l'instant et a l'avenir, la note, la niveau de difficulte, la vitenesse de jeu...

Définition à la ligne 20 du fichier gameManager.h.

4.4.2 Documentation des fonctions membres

4.4.2.1 unsigned int gameManager : :getScore ( ) [inline]

getScore : getter de la valeur courante du score

Renvoie

score : la note que le joueur a gagne

Définition à la ligne 59 du fichier gameManager.h.

4.4.2.2 Tetromino gameManager : :getTeromino ( ) [inline]

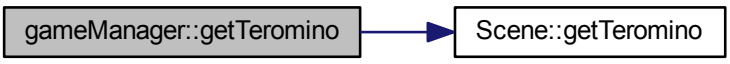
getTeromino getter du tetromino courant

Renvoie

S.getTeromino : le Tetromino courant(la piece qui tombe)

Définition à la ligne 68 du fichier gameManager.h.

Voici le graphe d'appel pour cette fonction :



4.4.2.3 void gameManager : :keyPressEvent (QKeyEvent * *ev*) [slot]

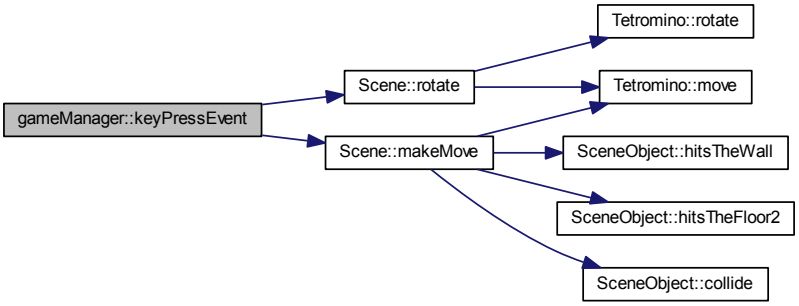
gestion des controles de clavier

Paramètres

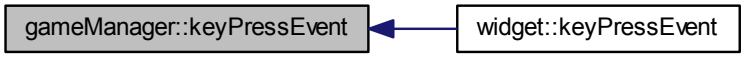
ev	: "Qt : :Key_Up" = retourner "Qt : :Key_Right" = bouger a droit "Qt : :Key_Down" = acceleration "Qt :Key_Left" = bouger a gauche

Définition à la ligne 91 du fichier gameManager.cpp.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.4.2.4 bool gameManager : :need_newTetromino ( ) [inline]

need_newTetromino

Renvoie

S.need_new_tetromino : true si c'est on si on a besoin un nouveau Tetromino

Définition à la ligne 77 du fichier gameManager.h.

4.4.2.5 void gameManager : :nextLevel (int) [signal]

nextLevel va emettre un signal quand on a un nouveau niveau

Paramètres

niveau<int>	: le nouveau niveau

Voici le graphe des appelants de cette fonction :

4.4.2.6 void gameManager : :scoreChanged (int) [signal]

scoreChanged va emettre un signal quand on a une nouvelle note

Paramètres

score<int>	: la nouvelle note

Voici le graphe des appelants de cette fonction :

4.4.2.7 QString gameManager : :Show ( ) [inline]

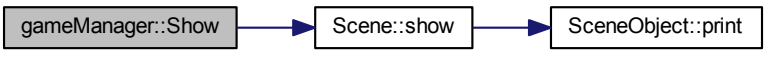
show : sert d'intermediaire entre la classe Scene et le widget d'affichage

Renvoie

renvoi "Game Over" si le jeu a termine, sinon elle renvoi le dernier etat en date de la scene du Tetris

Définition à la ligne 48 du fichier gameManager.h.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.4.2.8 void gameManager : :tetrominoChanged (int , int , int) [signal]

tetrominoChanged va emettre un signal quand on a un nouveau tetromino

Paramètres

tmpType<int>	: le type de tetromino
tmpRotate-Position<int>	: la postition de rotation de tetromino
tmp-Couleur<int>	: le couleur de tetromino

Voici le graphe des appelants de cette fonction :

4.4.2.9 void gameManager : :timerEvent (QTimerEvent * *event*) [protected]

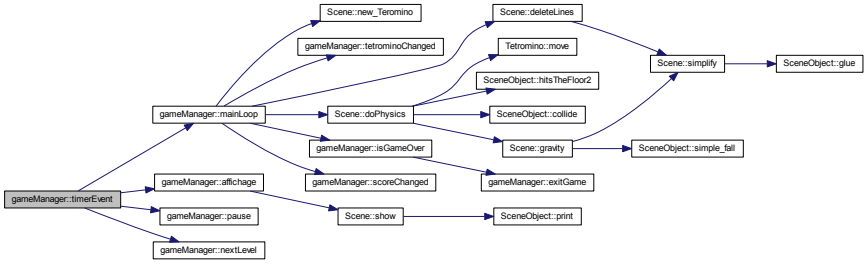
timerEvent va gerer les "timer event"

Paramètres

event	utiliser event->timerID pour verifier c'est le timeur qu'on veau : event->time.timerID : appeler mainloop() event->time2.timerID : appeler affichage() event->time3.timerID : gerer les niveaus

Définition à la ligne 131 du fichier gameManager.cpp.

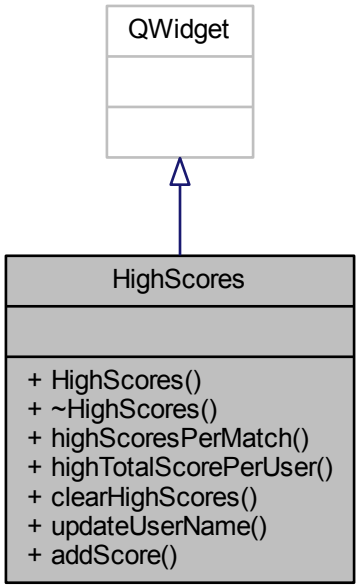
Voici le graphe d'appel pour cette fonction :



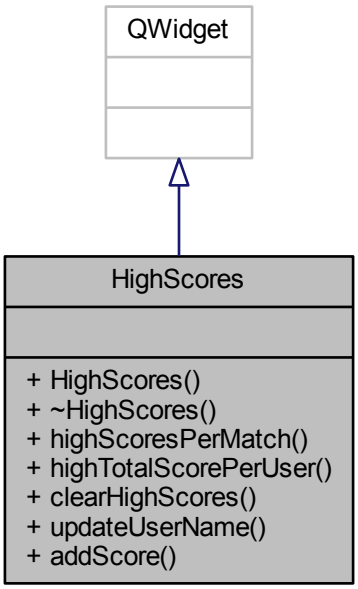
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/QTetrisCore/gameManager.h
– Source/QTetrisCore/gameManager.cpp

4.5 Référence de la classe HighScores

Graphe d'héritage de HighScores :



Graphe de collaboration de HighScores :



Connecteurs publics

– void **highScoresPerMatch** ()
– void **highTotalScorePerUser** ()
– void **clearHighScores** ()
– void **updateUserName** (QString)
– void **addScore** (int score)

Fonctions membres publiques

– **HighScores** (QWidget ∗parent=0)

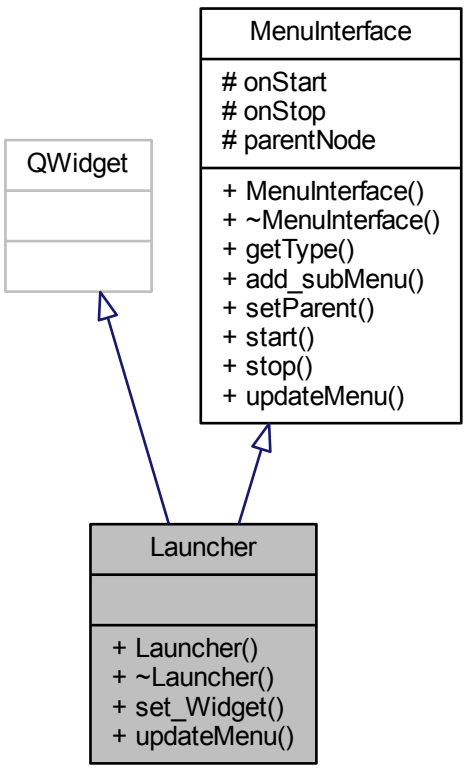
4.5.1 Description détaillée

Définition à la ligne 18 du fichier highScores.h.

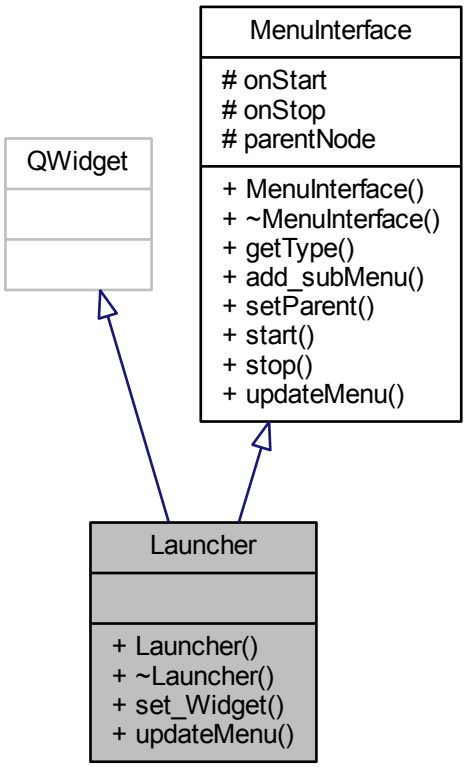
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ExtraWidgets/highScores.h
– Source/ExtraWidgets/highScores.cpp

4.6 Référence de la classe Launcher

Graphe d'héritage de Launcher :



Graphe de collaboration de Launcher :



Fonctions membres publiques

– **Launcher** (QWidget ∗parent=0, simpleMacro onStart_=[]{}, simpleMacro onStop_=[]{})
– void **set_Widget** (QWidget ∗widgetLaunched_)
– virtual void **updateMenu** ()

Additional Inherited Members

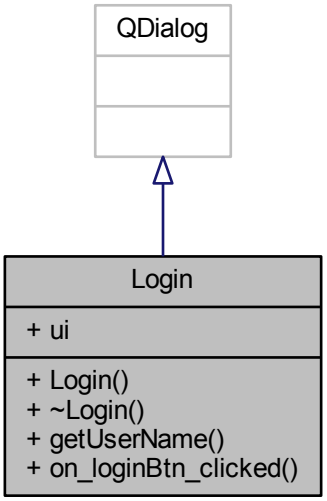
4.6.1 Description détaillée

Définition à la ligne 17 du fichier Launcher.h.

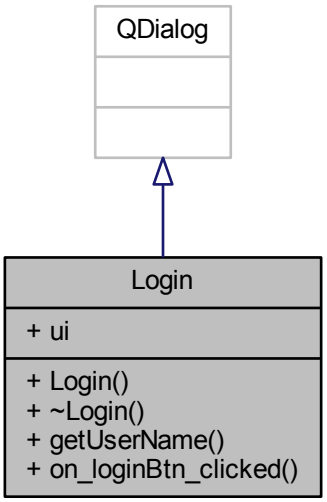
La documentation de cette classe a été générée à partir du fichier suivant :
– Source/MenuManager/Launcher.h

4.7 Référence de la classe Login

Graphe d'héritage de Login :



Graphe de collaboration de Login :



Connecteurs publics

– void **on_loginBtn_clicked** ()

Signaux

– void **logged** ()
– void **userNameChanged** (QString)

Fonctions membres publiques

– **Login** (QWidget ∗parent=0)
– QString **getUserName** ()

Attributs publics

– Ui : :Login ∗ **ui**

4.7.1 Description détaillée

Définition à la ligne 13 du fichier Login.h.

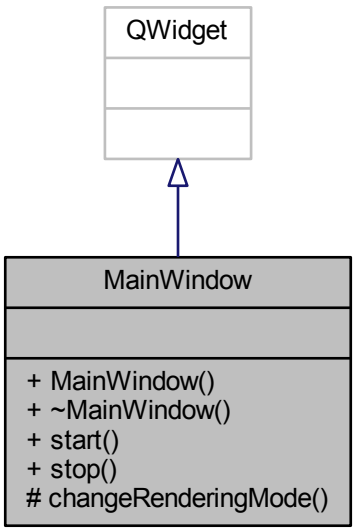
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ExtraWidgets/Login.h
– Source/ExtraWidgets/Login.cpp

4.8 Référence de la classe MainWindow

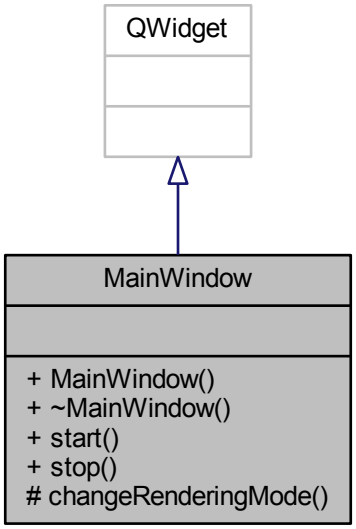
la fonction principaux du class mainwindow est a construire un GUI pour jour. Il va mettre en semble la widget de jeu, le widget de tetromino suivant, la note, le niveau, les choix des couleurs des pieces... Il va aussi gerer des connections entre les differents elements.

`#include <mainwindow.h>`

Graphe d'héritage de MainWindow :



Graphe de collaboration de MainWindow :



Connecteurs publics

– void start ()
quand le jeu viens de commencer, la fonction start va initiliser l'affichage de niveau et de note Il va aussi creer des connections entre les elements inclus.
– void stop ()
Guand la fonction stop recoit un signal de GameOver(), il va envoyer un signal de la note Il va aussi détruire des connections entre les elements inclus.

Signaux

– void new_Score (int score)
a la fin du jeu la fonction new_score va envoyer un signal au system de HighScores du jeu

Fonctions membres publiques

– **MainWindow** (QWidget ∗parent=0)

Connecteurs protégés

– void changeRenderingMode (int mode)
le slot changeRenderingMode va changer le type de rendering conrrespondant au signal

4.8.1 Description détaillée

la fonction principaux du class mainwindow est a construire un GUI pour jour. Il va mettre en semble la widget de jeu, le widget de tetromino suivant, la note, le niveau, les choix des couleurs des pieces... Il va aussi gerer des connections entre les differents elements.

Définition à la ligne 24 du fichier mainwindow.h.

4.8.2 Documentation des fonctions membres

4.8.2.1 void MainWindow : :changeRenderingMode (int *mode*) [protected], [slot]

le slot changeRenderingMode va changer le type de rendering conrrespondant au signal

Paramètres

mode	: le type de rendering mode = 0 : renderMode=SIMPLECOLOR mode = 1 : renderMode=GRADIENT mode = 2 : renderMode=TEXTURE

Définition à la ligne 76 du fichier mainwindow.cpp.

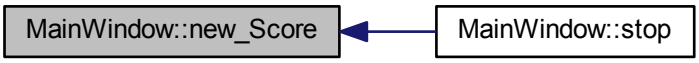
4.8.2.2 void MainWindow : :new_Score (int *score*) [signal]

a la fin du jeu la fonction new_score va envoyer un signal au system de HighScores du jeu

Paramètres

score	: le score obtenu du joueur

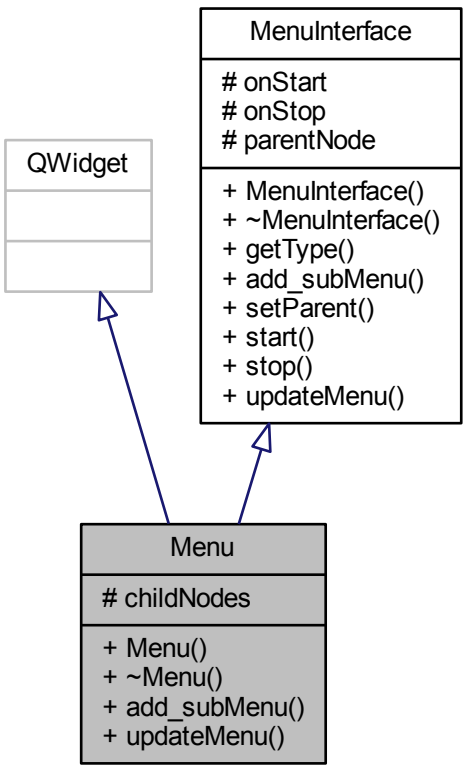
Voici le graphe des appelants de cette fonction :



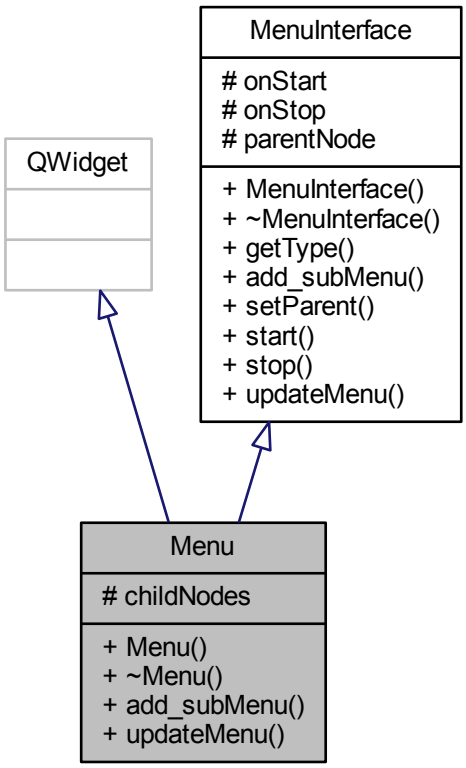
La documentation de cette classe a été générée à partir des fichiers suivants :
- Source/QTetrisCore/mainwindow.h
- Source/QTetrisCore/mainwindow.cpp

4.9 Référence de la classe Menu

Graphe d'héritage de Menu :



Graphe de collaboration de Menu :



Fonctions membres publiques

– **Menu** (QWidget ∗parent=0, simpleMacro onStart_=[]{}, simpleMacro onStop_=[]{})
– void **add_subMenu** (QString subMenu)
– virtual void **updateMenu** ()

Attributs protégés

– std : :vector< QString > **childNodes**

4.9.1 Description détaillée

Définition à la ligne 16 du fichier Menu.h.

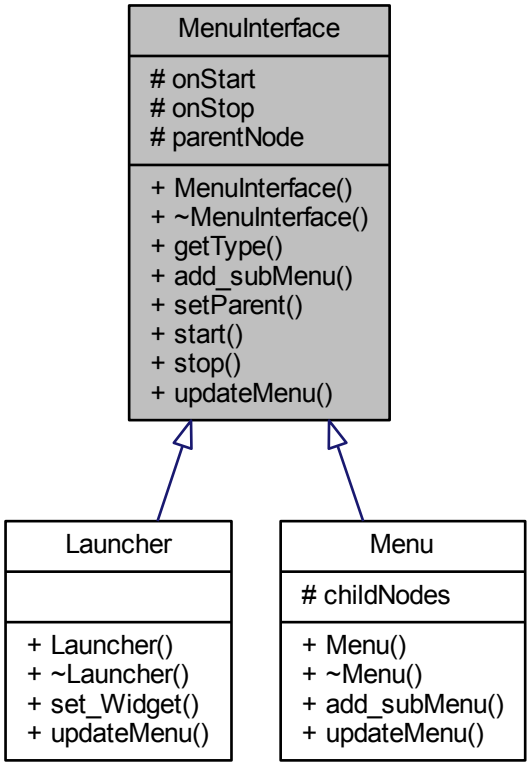
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/MenuManager/Menu.h
– Source/MenuManager/Menu.cpp

4.10 Référence de la classe MenuInterface

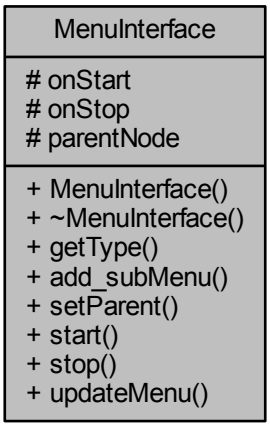
L'interface MenuInterface sert a regrouper les fonctionnalites communes ou pas de la classe Launcher et de la classe Menu. Elle sert a manipuler les deux classes filles indifferement et a limiter les redondances de code.

`#include <MenuInterface.h>`

Graphe d'héritage de MenuInterface :



Graphe de collaboration de MenuInterface :



Fonctions membres publiques

- **MenuInterface** (simpleMacro onStart_=[]{}, simpleMacro onStop_=[]{})
- virtual QString **getType** () const
- virtual void **add_subMenu** (QString subMenu)
- virtual void **setParent** (QString parentMenu)
- virtual void **start** ()
- virtual void **stop** ()
- virtual void **updateMenu** ()

Attributs protégés

- simpleMacro **onStart**
- simpleMacro **onStop**
- QString **parentNode**

4.10.1 Description détaillée

L'interface MenuInterface sert a regrouper les fonctionnalites communes ou pas de la classe Launcher et de la classe Menu. Elle sert a manipuler les deux classes filles indifferement et a limiter les redondances de code.

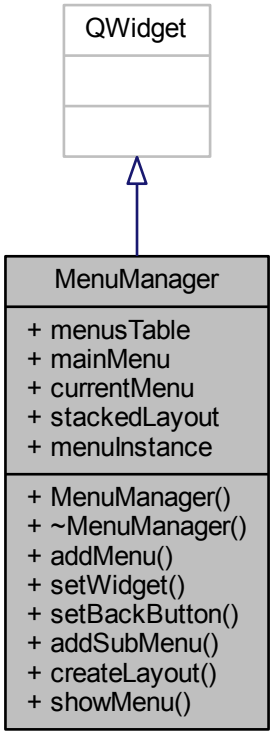
< fonctions lambdas a executer lors d'evenements particuliers

Définition à la ligne 16 du fichier MenuInterface.h.

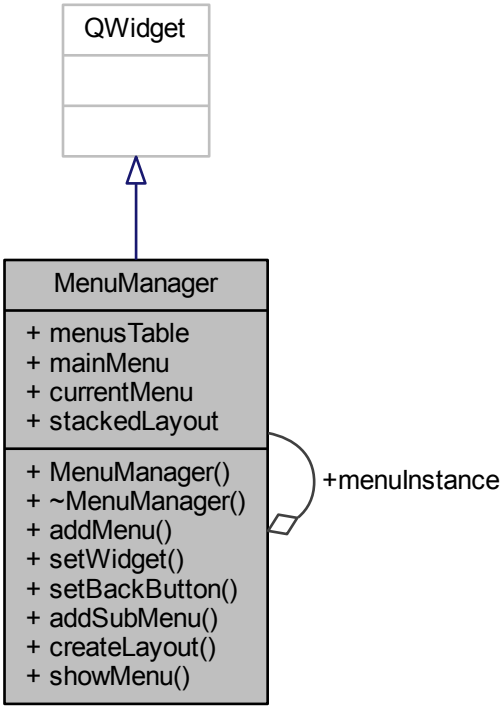
La documentation de cette classe a été générée à partir du fichier suivant :
- Source/MenuManager/MenuInterface.h

4.11 Référence de la classe MenuManager

Graphe d'héritage de MenuManager :



Graphe de collaboration de MenuManager :



Connecteurs publics

– void **showMenu** (QString newMenu)

Fonctions membres publiques

– **MenuManager** (QWidget ∗parent=0)
– MenuManager & **addMenu** (const QString &s, MenuInterface ∗m)
– MenuManager & **setWidget** (const QString &s, QWidget ∗w)
– MenuManager & **setBackButton** (const QString &s, const QString &b)
– MenuManager & **addSubMenu** (const QString &m, const QString &submenu)
– void **createLayout** ()

Attributs publics

– std : :map< QString,
std : :shared_ptr< MenuInterface > > **menusTable**
– QString **mainMenu**
– QString **currentMenu**
– QStackedLayout ∗ **stackedLayout**

Attributs publics statiques

– static MenuManager ∗ **menuInstance** =NULL

4.11.1 Description détaillée

Définition à la ligne 16 du fichier menumanager.h.

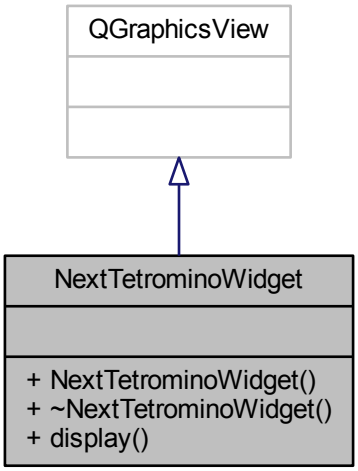
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/MenuManager/menumanager.h
– Source/MenuManager/menumanager.cpp

4.12 Référence de la classe NextTetrominoWidget

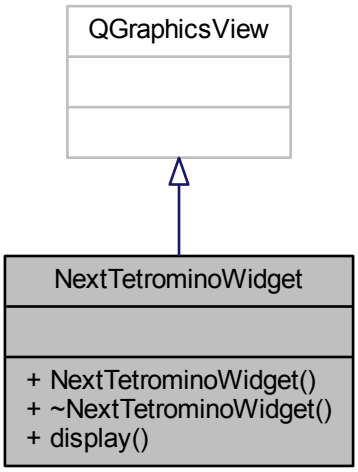
Cette classe est un objet qui peut etre present dans "scene" qui se trouve dans le coin de l'ecran. Il represente la piece a venir.

`#include <NextTetrominoWidget.h>`

Graphe d'héritage de NextTetrominoWidget :



Graphe de collaboration de NextTetrominoWidget :



Connecteurs publics

– void display (int, int, int)
display va affichager la piece suivante sur l'ecrain

Fonctions membres publiques

– **NextTetrominoWidget** (QWidget ∗parent=0)

4.12.1 Description détaillée

Cette classe est un objet qui peut etre present dans "scene" qui se trouve dans le coin de l'ecran. Il represente la piece a venir.

Définition à la ligne 16 du fichier NextTetrominoWidget.h.

4.12.2 Documentation des fonctions membres

4.12.2.1 void NextTetrominoWidget : :display (int *type,* int *rotationPosition,* int *couleur*) [slot]

display va affichager la piece suivante sur l'ecrain

Paramètres

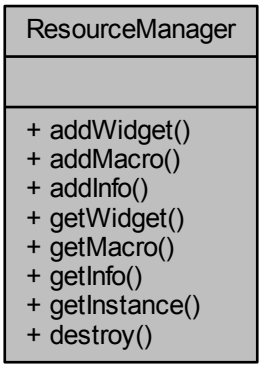
tmpType<int>	: le type de tetromino
tmpRotate-Position<int>	: la postition de rotation de tetromino
tmp-Couleur<int>	: le couleur de tetromino

Définition à la ligne 16 du fichier NextTetrominoWidget.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/QTetrisCore/NextTetrominoWidget.h
– Source/QTetrisCore/NextTetrominoWidget.cpp

4.13 Référence de la classe ResourceManager

Graphe de collaboration de ResourceManager :



Fonctions membres publiques

– ResourceManager & **addWidget** (const std : :string &s, QWidget ∗w)
– ResourceManager & **addMacro** (const std : :string &s, const simpleMacro &m)
– ResourceManager & **addInfo** (const QString &info, const QString &content)
– QWidget ∗ **getWidget** (const std : :string &s)
– simpleMacro **getMacro** (const std : :string &s)
– QString **getInfo** (QString info)

Fonctions membres publiques statiques

– static ResourceManager & **getInstance** ()
– static void **destroy** ()

4.13.1 Description détaillée

Définition à la ligne 13 du fichier ResourceManager.h.

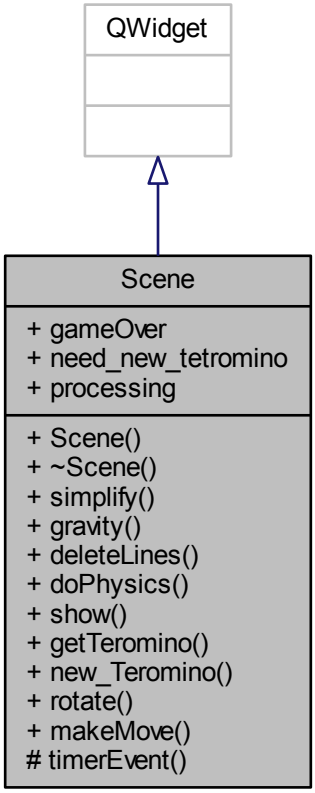
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ResourcesManager/ResourceManager.h
– Source/ResourcesManager/ResourceManager.cpp

4.14 Référence de la classe Scene

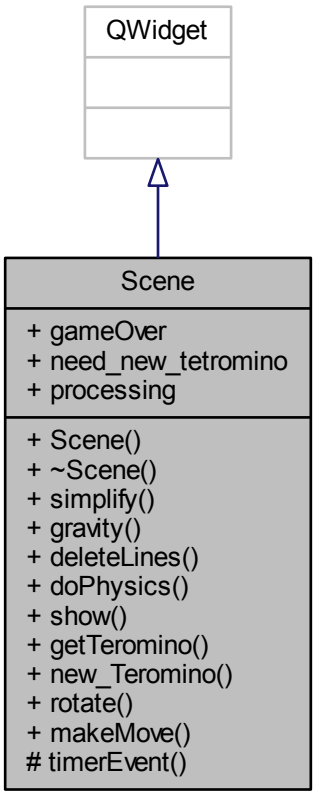
The Scene class is the scene manager used in the game. It's.

```
#include <Scene.h>
```

Graphe d'héritage de Scene :



Graphe de collaboration de Scene :



Fonctions membres publiques

– void simplify ()

simplify : regroupe les SceneObjects adjascents entre eux pour reduire le nombre d'objets separes presnets dans la scene.

– void gravity ()

gravity : fait tomber tout les objets(tout le monde tombe a la meme vitesse) et les relie entre eux (a l'aide de simplify())

– unsigned int deleteLines ()

deleteLines : suppression des lignes completes ; modifie le score

– void doPhysics ()

doPhysics : gere la physique (gravite+ suppressions de lignes multiples...etc)

– QString show ()

show : affichage de la carte

– Tetromino getTeromino () const

getTeromino : donne le tetromino courant qu'on manipule

– void new_Teromino (const TetrominoType &type, int rotation, int couleur)

new_Teromino

– void rotate ()

rotate fait tourner le tetromino courant.

– void makeMove (const Movement &commande)

makeMove : fait bouger le tetromino courant suivant la commande utilisateur (a droite, a gauche ou vers le bas). L'action ne sera pas effectuee si elle n'est pas valide.

Attributs publics

– bool **gameOver**
– bool **need_new_tetromino**
– bool **processing**

Fonctions membres protégées

– void timerEvent (QTimerEvent ∗event)

4.14.1 Description détaillée

The Scene class is the scene manager used in the game. It's.

Définition à la ligne 20 du fichier Scene.h.

4.14.2 Documentation des fonctions membres

4.14.2.1 unsigned int Scene : :deleteLines ( )

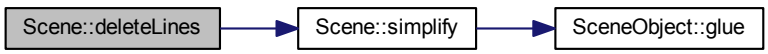
deleteLines : suppression des lignes completes ; modifie le score

Renvoie

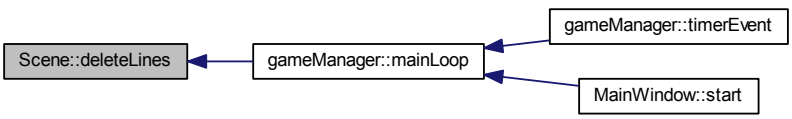
retourne le score

Définition à la ligne 135 du fichier Scene.cpp.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.14.2.2 Tetromino Scene : :getTeromino ( ) const

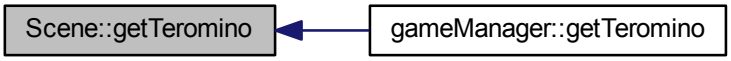
getTeromino : donne le tetromino courant qu'on manipule

Renvoie

le tetromino courant

Définition à la ligne 22 du fichier Scene.cpp.

Voici le graphe des appelants de cette fonction :



4.14.2.3 void Scene : :makeMove (const Movement & *commande*)

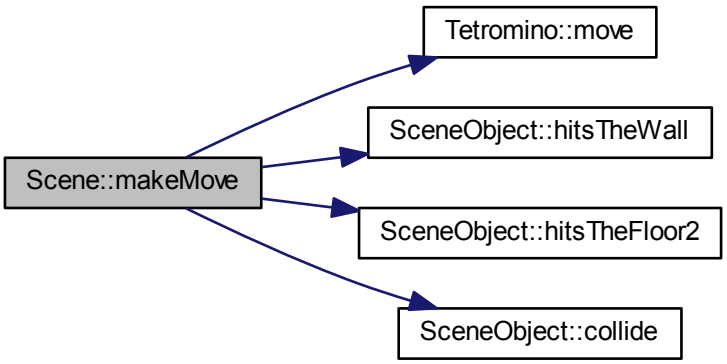
makeMove : fait bouger le tetromino courant suivant la commande utilisateur (a droite, a gauche ou vers le bas). L'action ne sera pas effectuee si elle n'est pas valide.

Paramètres

commande, :	la commande utilisateur

Définition à la ligne 280 du fichier Scene.cpp.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.14.2.4 void Scene : :new_Teromino (const TetrominoType & *type,* int *rotation,* int *couleur*)

new_Teromino

Paramètres

type	
rotation	
couleur	

Définition à la ligne 229 du fichier Scene.cpp.

Voici le graphe des appelants de cette fonction :

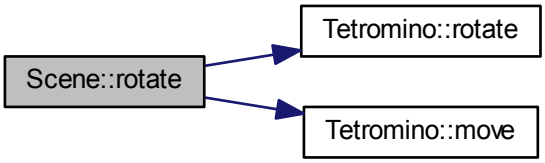
4.14.2.5 void Scene : :rotate ( )

rotate fait tourner le tetromino courant.

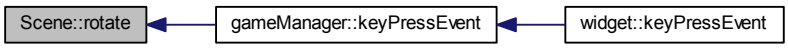
Si l'objet ne peut pas tourner car il a heurter un autre objet de la scene ou si il heurte un mur, l'objet sera decaler un peu a droite ou a gauche pour voir s'il peut s'y inserer. Sinon l'objet ne sera pas tourne car l'action invalide.

Définition à la ligne 235 du fichier Scene.cpp.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.14.2.6 QString Scene : :show ( )

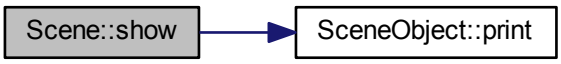
show : affichage de la carte

Renvoie

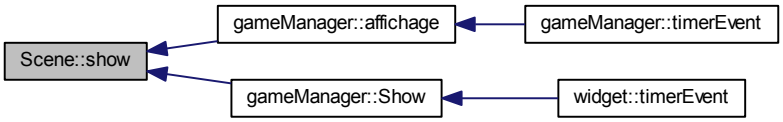
renvoie une chaine de caractere contenant l'ensemble

Définition à la ligne 27 du fichier Scene.cpp.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

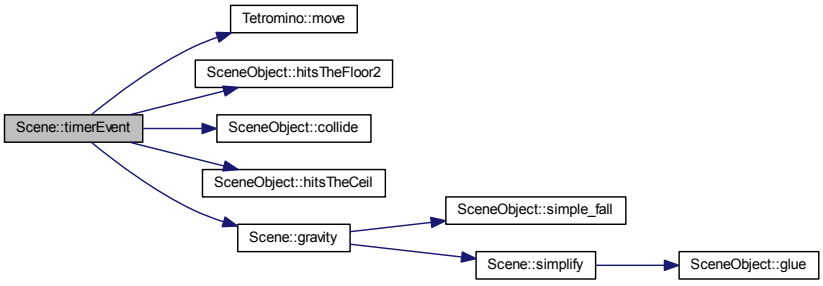


4.14.2.7 void Scene : :timerEvent (QTimerEvent * *event*) [protected]

< le mutex utilise par cette classe et le gamemanager pour ne pas generer d'incoherences entre les deux threads.

Définition à la ligne 196 du fichier Scene.cpp.

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :

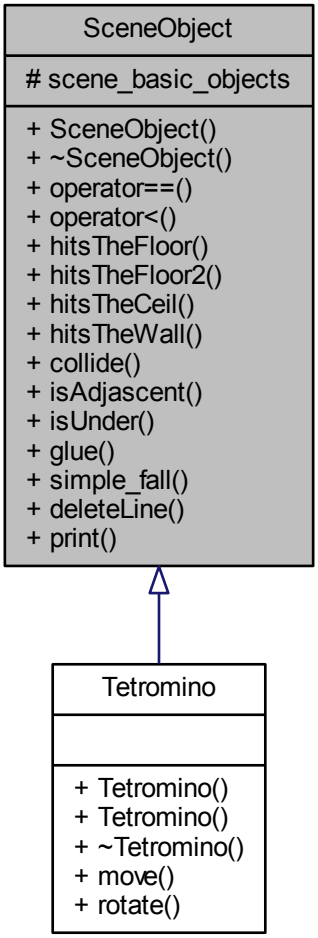
– Source/QTetrisCore/Scene.h
– Source/QTetrisCore/Scene.cpp

4.15 Référence de la classe SceneObject

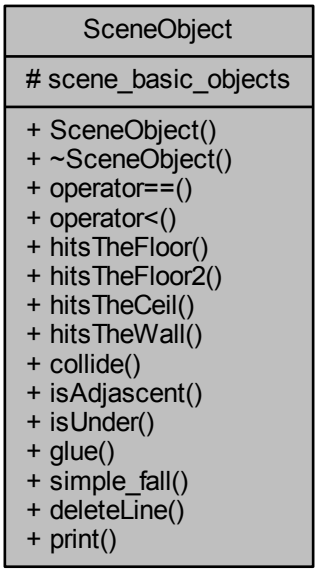
The SceneObject class.

`#include <SceneObject.h>`

Graphe d'héritage de SceneObject :



Graphe de collaboration de SceneObject :



Fonctions membres publiques

- **SceneObject** (int x, int y)
- bool operator== (const SceneObject &Obj) const
  Operateur de comparaison ==.
- bool operator< (const SceneObject &Obj) const
  Operateur de comparaison <.
- bool hitsTheFloor () const
  hitsTheFloor verifie si l'objet est arrive a la dernier ligne en bas. Typiquement, le SceneObject present sur la scene ne peut plus descendre plus bas quand cette methode retourne true.
- bool hitsTheFloor2 () const
  hitsTheFloor2 verifie si l'objet est enfonce en bas de la scene.
- bool hitsTheCeil () const
  hitsTheCeil verifie si l'objet est enfonce en haut de la scene.
- bool hitsTheWall () const
  hitsTheWall
- bool collide (const SceneObject &Obj) const
  collide : verifie la collision entre deux SceneObject, c'est a dire les deux possedent des parties qui occupent la meme zone spaciale(meme x et y) sur la scene.
- bool isAdjascent (const SceneObject &Obj) const
  isAdjascent verifie si les deux SceneObject sont adjascents, c'est a dire si l'un deux possede une partie qui est directement a droite de l'autre(ou a sa gauche...).
- bool isUnder (const SceneObject &Obj) const
  isUnder verifie si l'objet courant possede une partie qui est directement en dessus de Obj.
- void glue (SceneObject Obj)
  glue : comme le nom l'indique, cette fonction colle Obj a l'object courant. Rien n'interdit de coller deux SceneObject, pourvu qu'ils soient disjoints, mais pour preserver la logique du jeu, il ne faut coller que les objets proches entre eux(qui peuvent former un bloc unique).
- SceneObject simple_fall () const
  simple_fall : fait tomber l'objet courant d'une ligne
- std : :tuple< std : :set
  < SceneObject >, bool > deleteLine (int l) const
  deleteLine : verifie supprime la ligne l si celle-ci peut etre complete.
- void print (char **map) const

print : affiche le contenu du SceneObject courant sur le tableau passe en parametre

Attributs protégés

– std : :set< Points > **scene_basic_objects**

4.15.1 Description détaillée

The SceneObject class.

Définition à la ligne 57 du fichier SceneObject.h.

4.15.2 Documentation des fonctions membres

4.15.2.1 bool SceneObject : :collide (const SceneObject & *Obj*) const

collide : verifie la collision entre deux SceneObject, c'est a dire les deux possedent des parties qui occupent la meme zone spaciale(meme x et y) sur la scene.

Paramètres

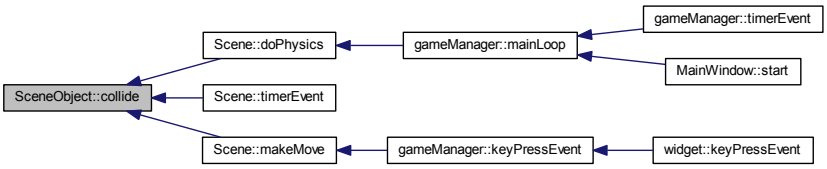
Obj, :	l'autre SceneObject a comparer avec le SceneObjet courant

Renvoie

true s'il y a collision.

Définition à la ligne 145 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



4.15.2.2 std : :tuple< std : :set< SceneObject >, bool > SceneObject : :deleteLine (int *l*) const

deleteLine : verifie supprime la ligne l si celle-ci peut etre complete.

Apres suppression, on obtient des fragments connexes qui composent l'objet courant apres suppression de la ligne.

Paramètres

l, :	la ligne a supprimer

Renvoie

les fragments connexes de l'objet courant apress suppression+ un boolean qui est egale a true s'il y a eu suppression

Définition à la ligne 232 du fichier SceneObject.cpp.

4.15.2.3 void SceneObject : :glue (SceneObject *Obj*)

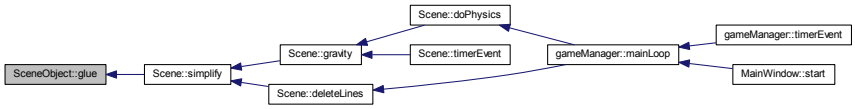
glue : comme le nom l'indique, cette fonction colle Obj a l'object courant. Rien n'interdit de coller deux SceneObject, pourvu qu'ils soient disjoints, mais pour preserver la logique du jeu, il ne faut coller que les objets proches entre eux(qui peuvent former un bloc unique).

Paramètres

Obj, :	le SceneObjet a coller avec l'object courant

Définition à la ligne 208 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



4.15.2.4 bool SceneObject : :hitsTheCeil ( ) const

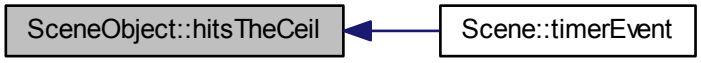
hitsTheCeil verifie si l'objet est enfonce en haut de la scene.

Renvoie

true si une partie de l'SceneObject courant depasse le haut de la scene

Définition à la ligne 122 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



4.15.2.5 bool SceneObject : :hitsTheFloor ( ) const

hitsTheFloor verifie si l'objet est arrive a la dernier ligne en bas. Typiquement, le SceneObject present sur la scene ne peut plus descendre plus bas quand cette methode retourne true.

Renvoie

true si l'objet atteint la derniere ligne du bas

Définition à la ligne 100 du fichier SceneObject.cpp.

4.15.2.6 bool SceneObject : :hitsTheFloor2 ( ) const

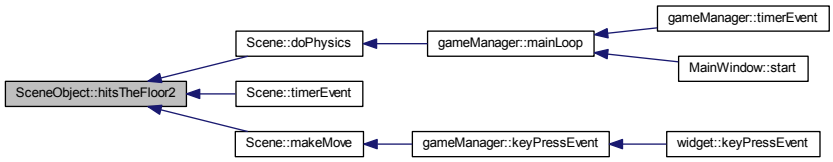
hitsTheFloor2 verifie si l'objet est enfonce en bas de la scene.

Renvoie

true si une partie de l'SceneObject courant depasse le bas de la scene

Définition à la ligne 111 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



4.15.2.7 bool SceneObject : :hitsTheWall ( ) const

hitsTheWall

Renvoie

true si une partie de l'SceneObject courant depasse la scene de l'un des deux cotes.

Définition à la ligne 133 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :

4.15.2.8 bool SceneObject : :isAdjascent (const SceneObject & *Obj*) const

isAdjascent verifie si les deux SceneObject sont adjascents, c'est a dire si l'un deux possede une partie qui est directement a droite de l'autre(ou a sa gauche...).

Paramètres

Obj, :	l'autre SceneObject a comparer avec le SceneObjet courant

Renvoie

true si Obj et l'objet courant sont adjascents

Définition à la ligne 164 du fichier SceneObject.cpp.

4.15.2.9 bool SceneObject : :isUnder (const SceneObject & *Obj*) const

isUnder verifie si l'objet courant possede une partie qui est directement en dessus de Obj.

Paramètres

Obj, :	l'autre SceneObject a comparer avec le SceneObjet courant

Renvoie

retourn true si l'objet courant possede une partie qui est directement en dessus de Obj

Définition à la ligne 186 du fichier SceneObject.cpp.

4.15.2.10 bool SceneObject : :operator< (const SceneObject & *Obj*) const

Operateur de comparaison <.

Par definition(le fonctionnement intrinseque de la classe Scene qui stock les SceneObject), les SceneObjects qu'on manipule sont disjoints. On peut alors creer une relation d'ordre en comparant les cubes elementaires les plus grands de Obj et de l'objet courant.

Paramètres

Obj	

Renvoie

true si Obj est plus petit le l'objet courant

Définition à la ligne 72 du fichier SceneObject.cpp.

4.15.2.11 bool SceneObject : :operator== (const SceneObject & *Obj*) const

Operateur de comparaison ==.

Paramètres

Obj, :	un autre SceneObject a comparer avec l'objet courant

Renvoie

true si Obj est egale a l'object courante (point par point) sans prendre en consideration des couleurs

Définition à la ligne 57 du fichier SceneObject.cpp.

4.15.2.12 void SceneObject : :print (char ** *map*) const

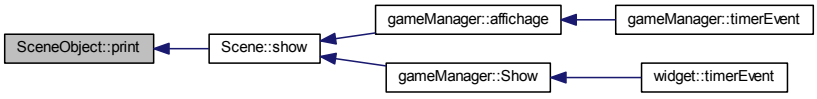
print : affiche le contenu du SceneObject courant sur le tableau passe en parametre

Paramètres

map	tableau de taille [Size_x][Size_y]

Définition à la ligne 223 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



4.15.2.13 SceneObject SceneObject : :simple_fall ( ) const

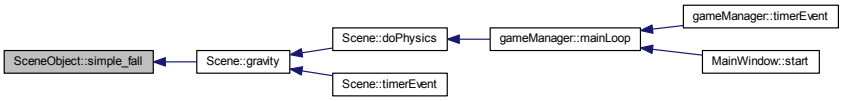
simple_fall : fait tomber l'objet courant d'une ligne

Renvoie

la copie de l'objet courant translate vers le bas.

Définition à la ligne 215 du fichier SceneObject.cpp.

Voici le graphe des appelants de cette fonction :



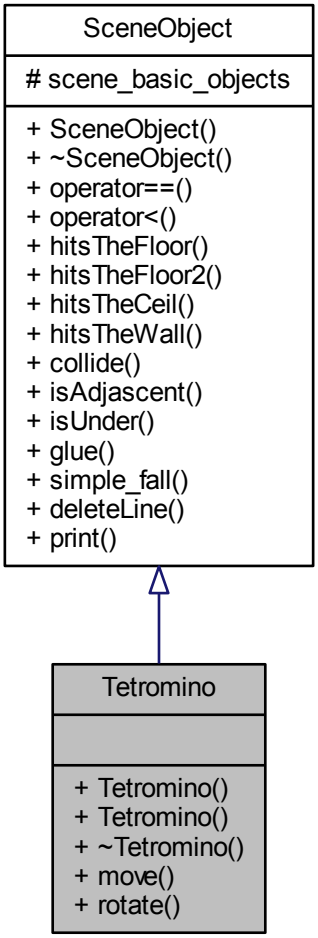
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/QTetrisCore/SceneObject.h
– Source/QTetrisCore/SceneObject.cpp

4.16 Référence de la classe Tetromino

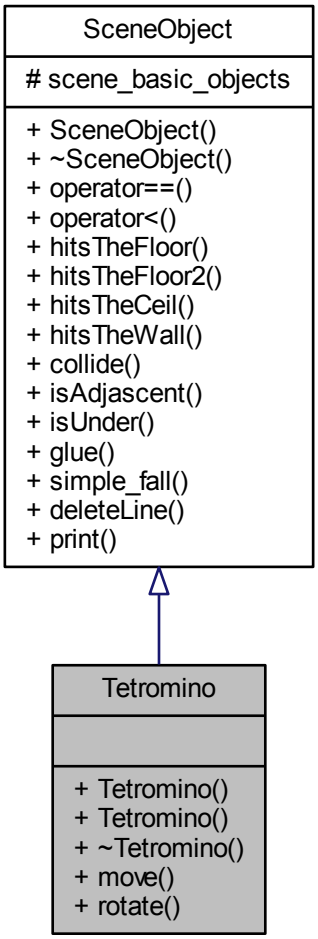
La classe Tetromino est un objet qui peut etre lui aussi present dans la Scene. Il represente la piece tombante que l'utilisateur peut controller.

`#include <Tetromino.h>`

Graphe d'héritage de Tetromino :



Graphe de collaboration de Tetromino :



Fonctions membres publiques

– Tetromino (const TetrominoType &type, const int rotatePosition, int couleur)
  Constructeur Tetromino.
– Tetromino move (const Movement &commande) const
  move
– Tetromino rotate () const
  rotate

Additional Inherited Members

4.16.1 Description détaillée

La classe Tetromino est un objet qui peut etre lui aussi present dans la Scene. Il represente la piece tombante que l'utilisateur peut controller.

Définition à la ligne 23 du fichier Tetromino.h.

4.16.2 Documentation des constructeurs et destructeur

4.16.2.1 Tetromino : :Tetromino (const TetrominoType & *type,* const int *rotatePosition,* int *couleur*)

Constructeur Tetromino.

Paramètres

type	: l'entier qui determine le type de tetromino(L,T,Z...). Il s'agit du premier indice du tableau tetrominoDatabase. Il varie donc entre 0 et 6.
rotatePosition	: l'entier qui determine la rotation de tetromino(L,T,Z...). Il s'agit du deuxieme indice du tableau tetrominoDatabase. Il varie entre 0 et 3.
couleur	: L'entier qui determine la couleur. Il s'agit de l'indice de colorTab.

Définition à la ligne 14 du fichier Tetromino.cpp.

4.16.3 Documentation des fonctions membres

4.16.3.1 Tetromino Tetromino : :move (const Movement & *commande*) const

move

Cree une copie de l'objet avec l'une transformation du type To_LEFT,...

Paramètres

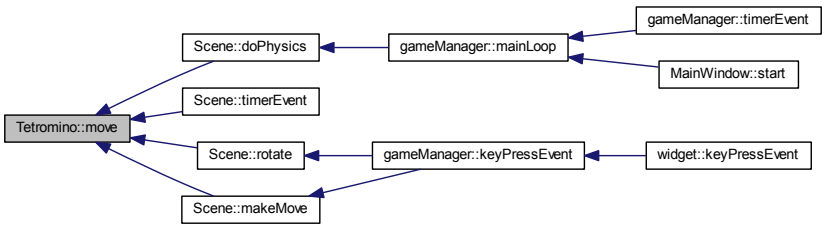
commande, :	la transformation à appliquer à l'objet courant.

Renvoie

la copie transformée de l'objet courant.

Définition à la ligne 48 du fichier Tetromino.cpp.

Voici le graphe des appelants de cette fonction :



4.16.3.2 Tetromino Tetromino : :rotate ( ) const

rotate

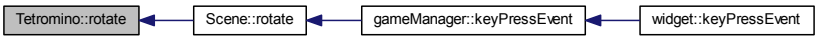
Cree une copie de l'objet avec une rotation comme transformation.

Renvoie

la copie transformée de l'objet courant.

Définition à la ligne 30 du fichier Tetromino.cpp.

Voici le graphe des appelants de cette fonction :



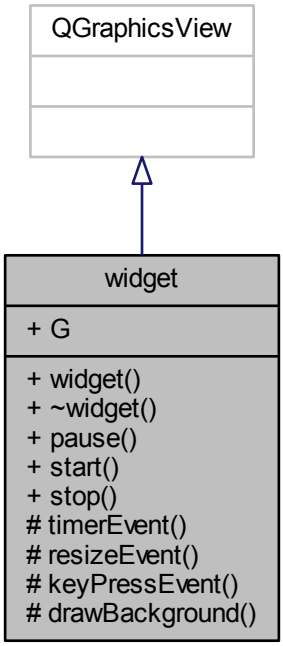
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/QTetrisCore/Tetromino.h
– Source/QTetrisCore/Tetromino.cpp

4.17 Référence de la classe widget

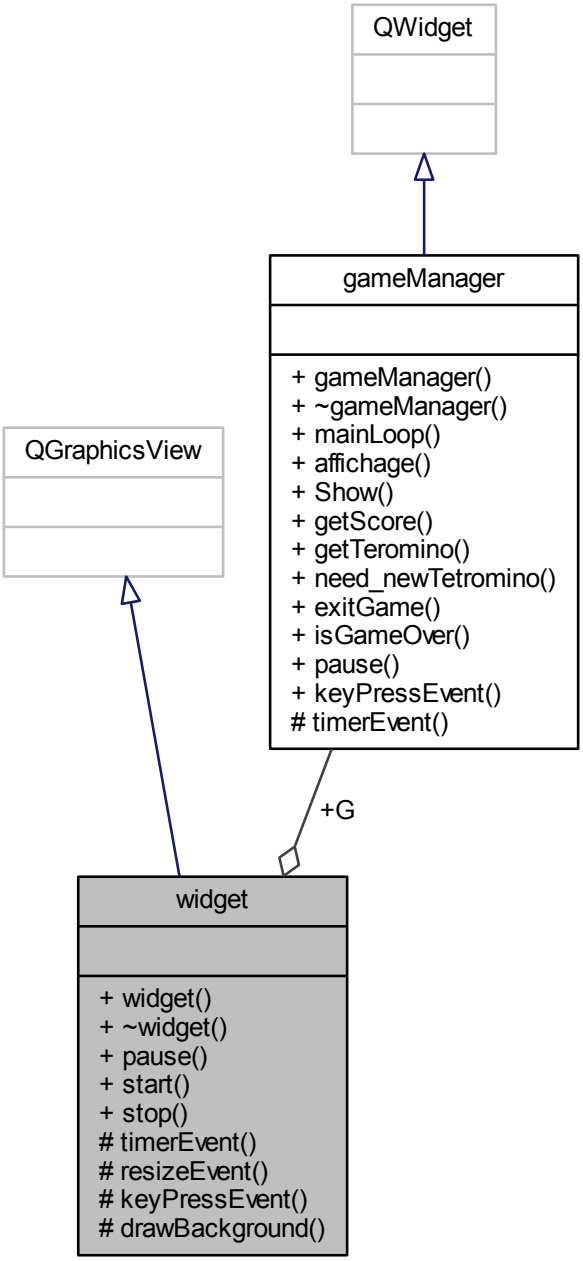
Affichager le scene de jeu et traiter préliminairement les controles qui vont etre passe au gameManager.

`#include <widget.h>`

Graphe d'héritage de widget :



Graphe de collaboration de widget :



Connecteurs publics

– void pause ()

pause ou continue le jeu (selon le choix de l'utilisateur)

– void start ()

mettre en service le widget

– void stop ()

arreter le widget

Signaux

– void gameOver ()
emettre le signal de gameOver quand le jeu est termine

Fonctions membres publiques

– **widget** (QWidget ∗parent=0)

Attributs publics

– gameManager ∗ **G**

Fonctions membres protégées

– void timerEvent (QTimerEvent ∗event)
timerEvent gere l'affichage des pixels dans la scene
– void resizeEvent (QResizeEvent ∗event)
resizeEvent adapte la taille de "scene", sans lui la scene est trop petite.
– void keyPressEvent (QKeyEvent ∗ev)
quand le timer et gameManager marche, keyPressEvent() va passer les evenements au GameManager sinon il va passer le controle au parent
– void drawBackground (QPainter ∗painter, const QRectF &rect)
fonction drawBackground va affichager le fond de scene. Cette fonction est geree par Qt.

4.17.1 Description détaillée

Affichager le scene de jeu et traiter préliminairement les controles qui vont etre passe au gameManager.

Définition à la ligne 19 du fichier widget.h.

4.17.2 Documentation des fonctions membres

4.17.2.1 void widget : :keyPressEvent (QKeyEvent ∗ *ev*) [protected]

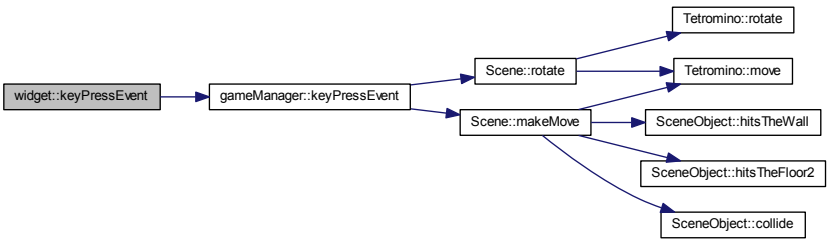
quand le timer et gameManager marche, keyPressEvent() va passer les evenements au GameManager sinon il va passer le controle au parent

Paramètres

ev	: ev signifie le type event, e.g. Qt : :Key_Up Qt : :Key_Right Qt : :Key_Down Qt : :Key_Left

Définition à la ligne 79 du fichier widget.cpp.

Voici le graphe d'appel pour cette fonction :



4.17.2.2 void widget : :timerEvent (QTimerEvent ∗ *event*) [protected]

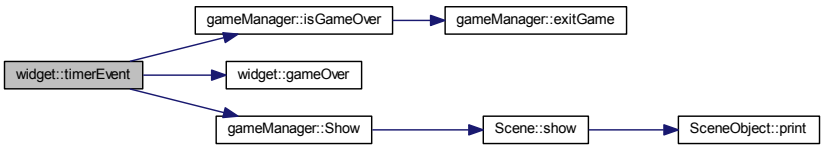
timerEvent gere l'affichage des pixels dans la scene

Paramètres

event	: utiliser event->timerID pour verifier c'est le timeur qu'on veau

Définition à la ligne 18 du fichier widget.cpp.

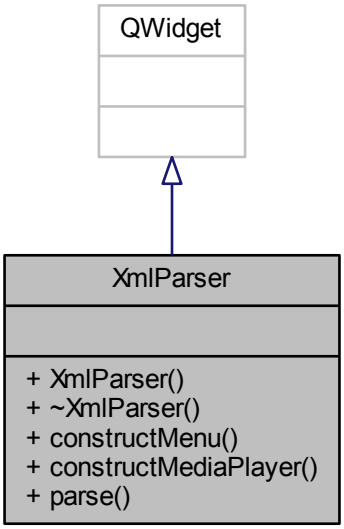
Voici le graphe d'appel pour cette fonction :



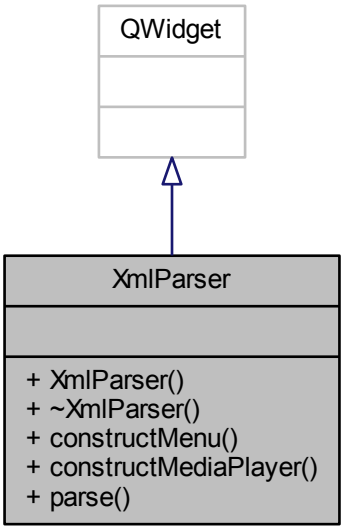
La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/QTetrisCore/widget.h
– Source/QTetrisCore/widget.cpp

4.18 Référence de la classe XmlParser

Graphe d'héritage de XmlParser :



Graphe de collaboration de XmlParser :



Fonctions membres publiques

– **XmlParser** (QWidget ∗parent=0)
– void **constructMenu** (QDomNode n, const QString &s)
– void **constructMediaPlayer** (QDomNode n)
– bool **parse** ()

4.18.1 Description détaillée

Définition à la ligne 8 du fichier XmlParser.h.

La documentation de cette classe a été générée à partir des fichiers suivants :
– Source/ResourcesManager/XmlParser.h
– Source/ResourcesManager/XmlParser.cpp

Index