Qt基本功能

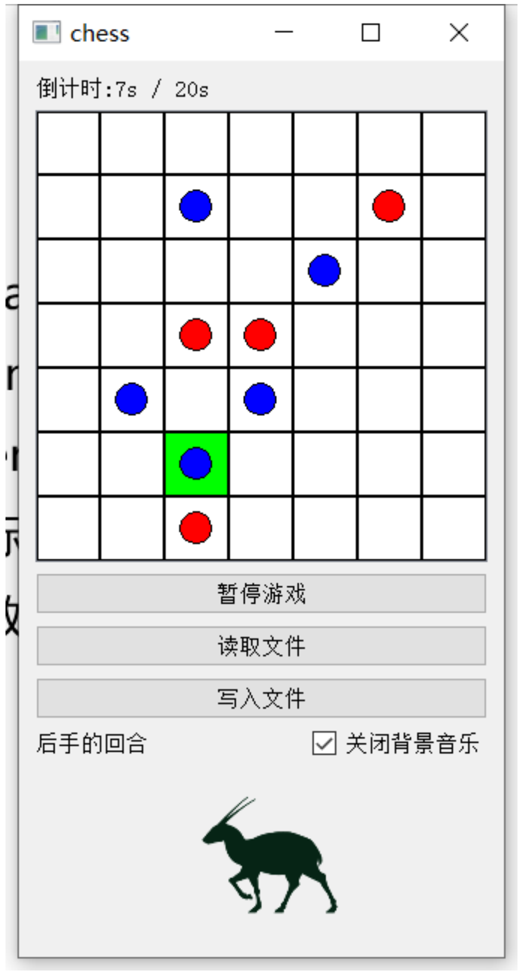
荣易

2022.08.24

https://github.com/glassesq/chess

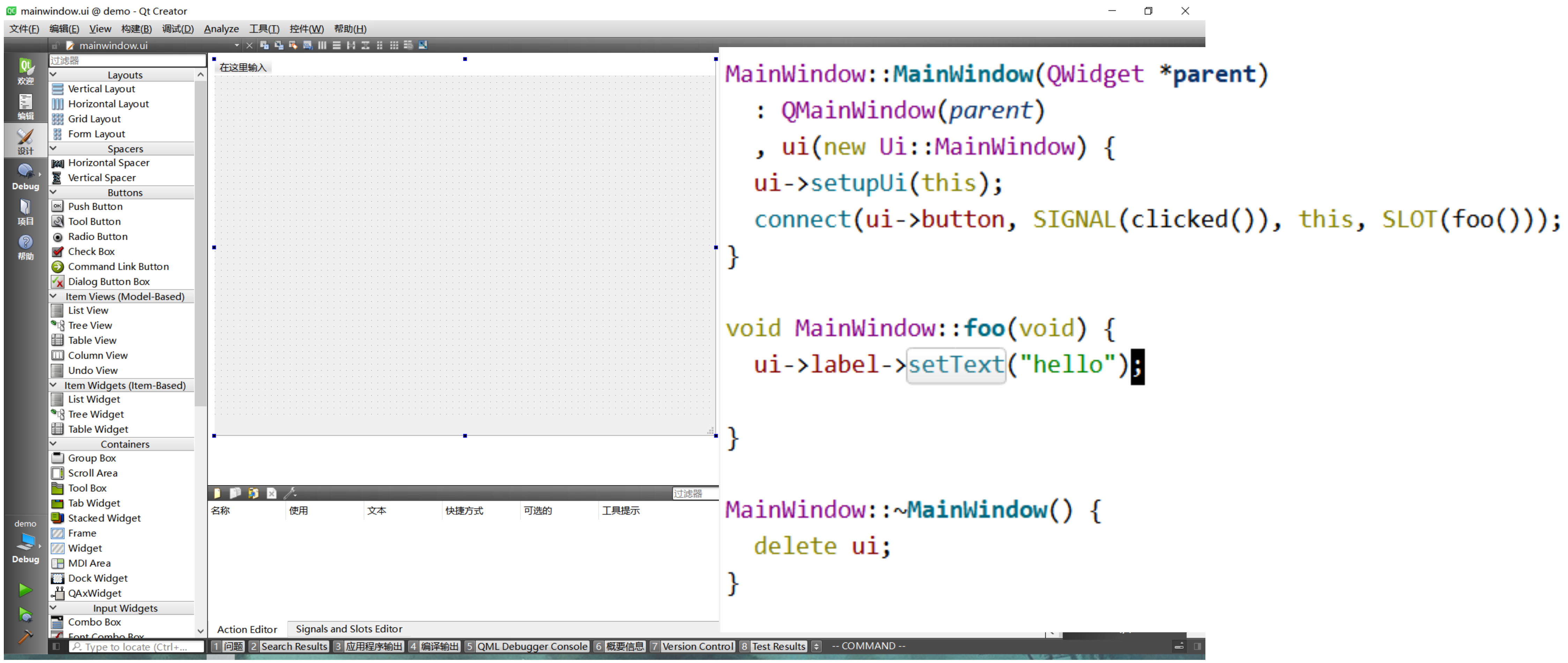
大纲

- widget / layout
- 事件
- slots / signal
- view / scene
- 键盘 / 鼠标事件 / 事件过滤器
- 音乐 / 音效播放
- 简易动画
- 文件读取
- windeployqt



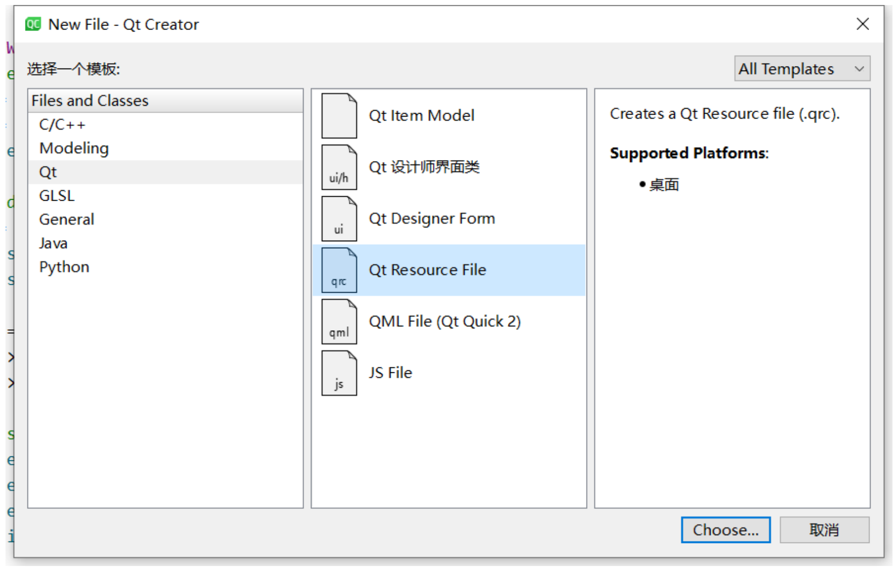
Qt文档：https://doc.qt.io/qt-5.15/classes.html

可视化设计



```
MainWindow::MainWindow(QWidget *parent)
  : QMainWindow(parent)
  , ui(new Ui::MainWindow) {
  ui->setupUi(this);
  connect(ui->button, SIGNAL(clicked()), this, SLOT(foo()));
}

void MainWindow::foo(void) {
  ui->label->setText("hello");

}

MainWindow::~MainWindow() {
  delete ui;
}
```

资源文件



New File - Qt Creator

选择一个模板:

All Templates

Files and Classes
- C/C++
- Modeling
- Qt
- GLSL
- General
- Java
- Python

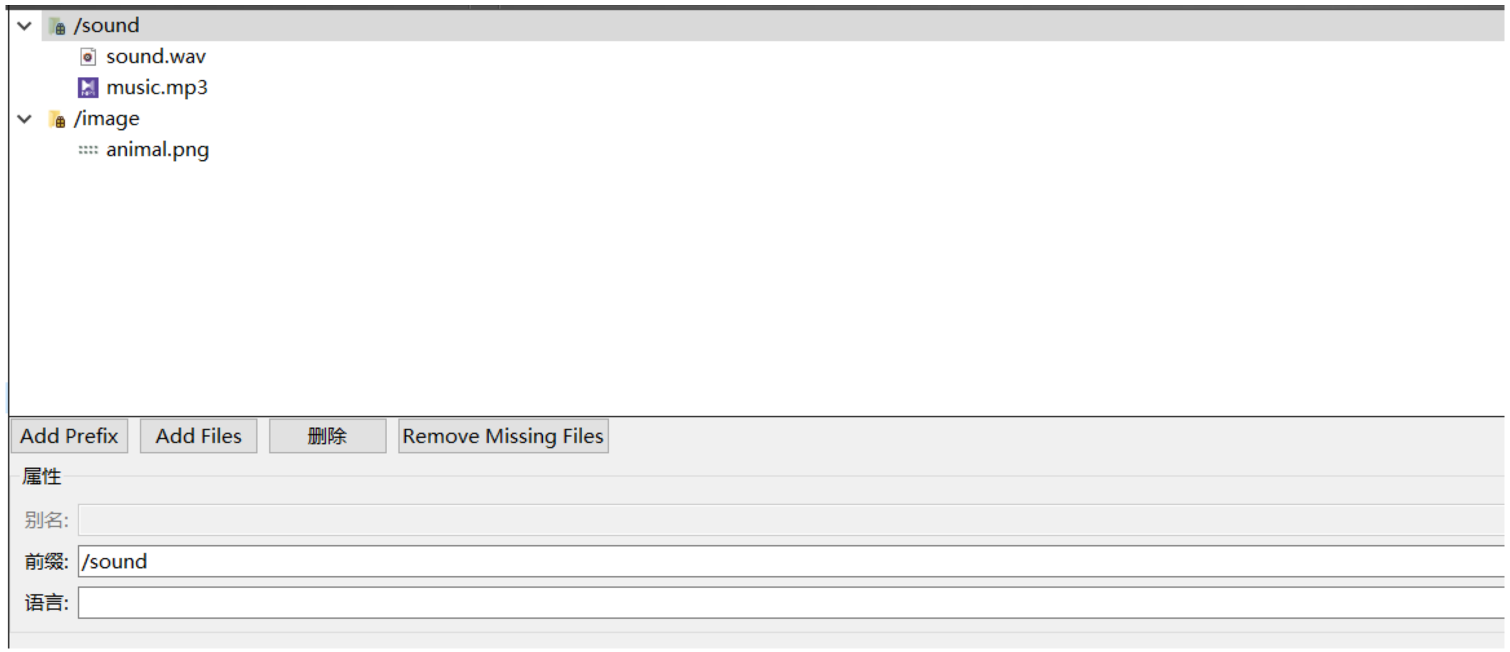
- Qt Item Model
- Qt 设计师界面类
- Qt Designer Form
- Qt Resource File
- QML File (Qt Quick 2)
- JS File

Creates a Qt Resource file (.qrc).

Supported Platforms:

- 桌面

Choose... 取消



- /sound
  - sound.wav
  - music.mp3
- /image
  - animal.png

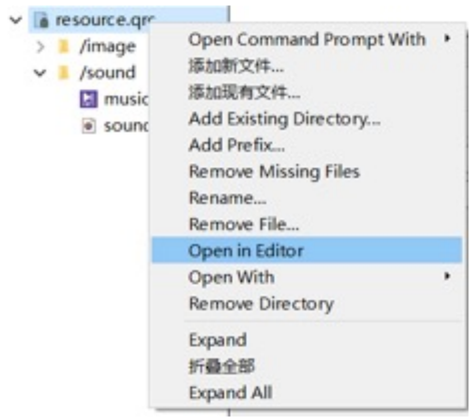
Add Prefix | Add Files | 删除 | Remove Missing Files

属性

别名:

前缀: /sound

语言:



- resource.qrc
  - /image
  - /sound
    - music
    - sound

widget / layout

```cpp
/* layout */
QVBoxLayout *layout = new QVBoxLayout();
layout->addWidget(timeboard);
layout->addWidget(view);
layout->addWidget(button, Qt::AlignCenter);

layout->addWidget(readfile);
layout->addWidget(writefile);

QHBoxLayout *hlayout = new QHBoxLayout();
hlayout->addWidget(message, Qt::AlignRight);
hlayout->addWidget(checkbox, Qt::AlignLeft);

layout->addLayout(hlayout);
```

```cpp
widget = new QWidget();
widget->setLayout(layout);
widget->installEventFilter(this);
widget->setFocusPolicy(Qt::NoFocus);
setCentralWidget(widget);
```

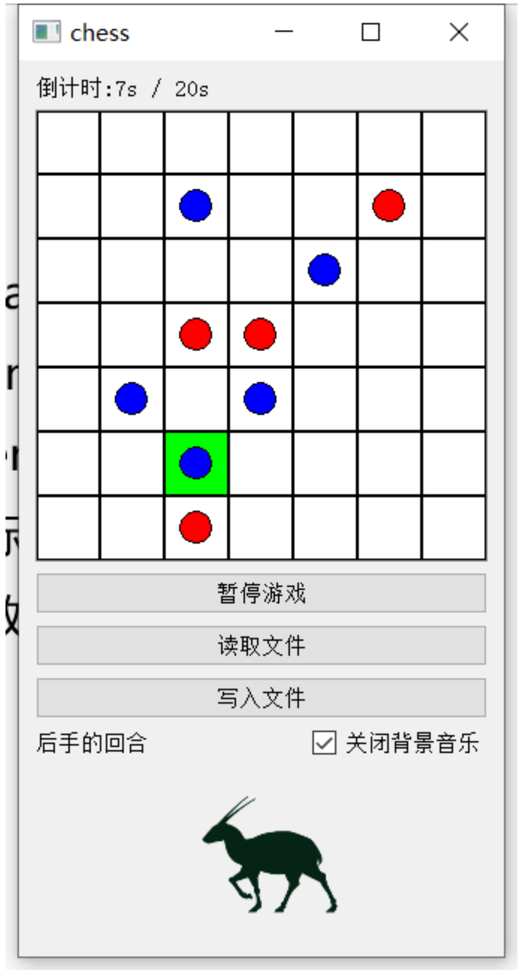
```cpp
QLabel* timeboard;
```

```cpp
QGraphicsScene* scene;
QGraphicsView* view;
```

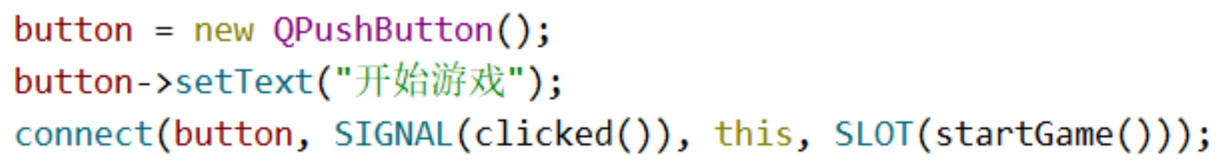
```cpp
QPushButton* button;
QPushButton* readfile;
QPushButton* writefile;
```

```cpp
QLabel* message;     QCheckBox* checkbox;
```

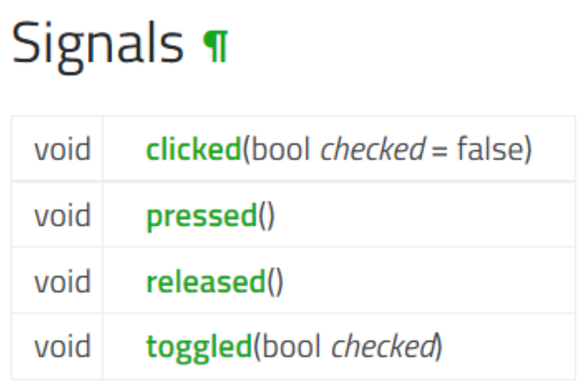
```cpp
Ani* ani;
class Ani : public QWidget {
```



slots / signals

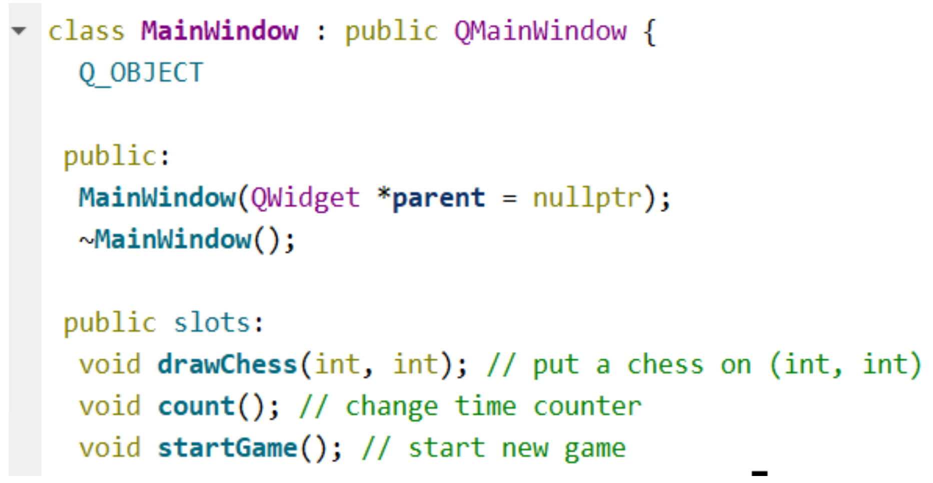


```
button = new QPushButton();
button->setText("开始游戏");
connect(button, SIGNAL(clicked()), this, SLOT(startGame()));
```

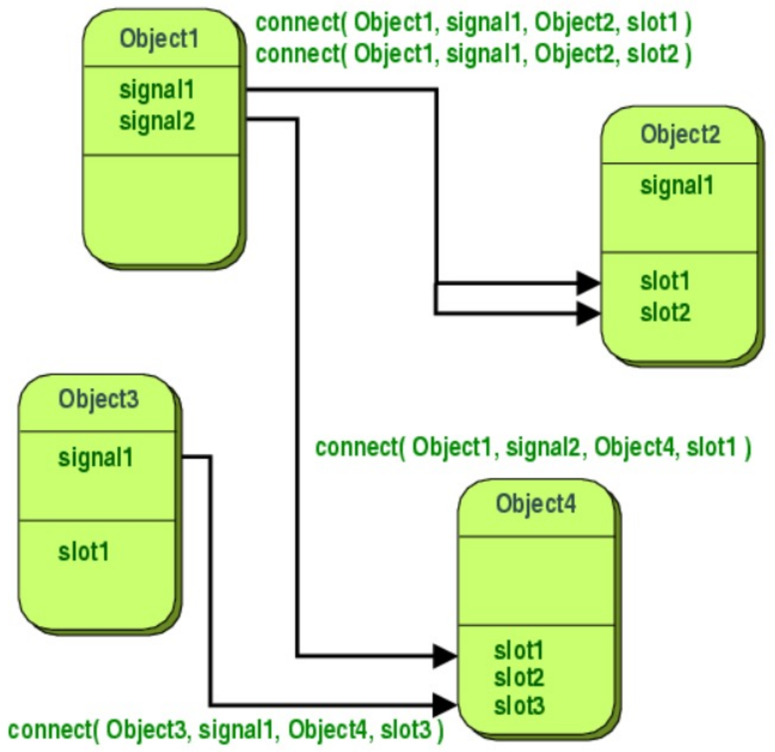


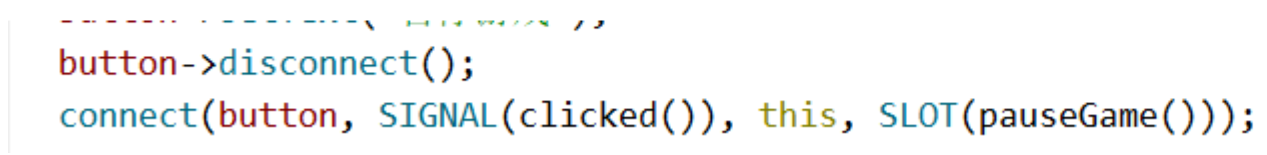
Signals ¶

void	clicked(bool *checked* = false)
void	pressed()
void	released()
void	toggled(bool *checked*)



```
class MainWindow : public QMainWindow {
  Q_OBJECT

 public:
  MainWindow(QWidget *parent = nullptr);
  ~MainWindow();

 public slots:
  void drawChess(int, int); // put a chess on (int, int)
  void count(); // change time counter
  void startGame(); // start new game
```


Object1
signal1
signal2
connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

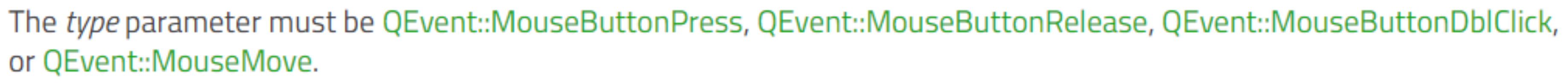


```
[illegible]
button->disconnect();
connect(button, SIGNAL(clicked()), this, SLOT(pauseGame()));
```

https://doc.qt.io/qt-6/signalsandslots.html

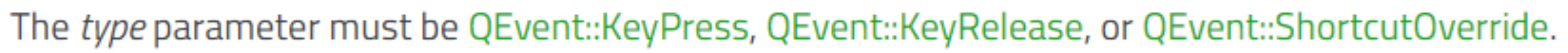
事件

- **事件：** “发生的事”
- 事件的类（class）与事件类型（type）
    - 获取事件类型：event->type();
    - QMouseEvent(class):



The *type* parameter must be QEvent::MouseButtonPress, QEvent::MouseButtonRelease, QEvent::MouseButtonDblClick, or QEvent::MouseMove.

    - QKeyEvent(class):



The *type* parameter must be QEvent::KeyPress, QEvent::KeyRelease, or QEvent::ShortcutOverride.

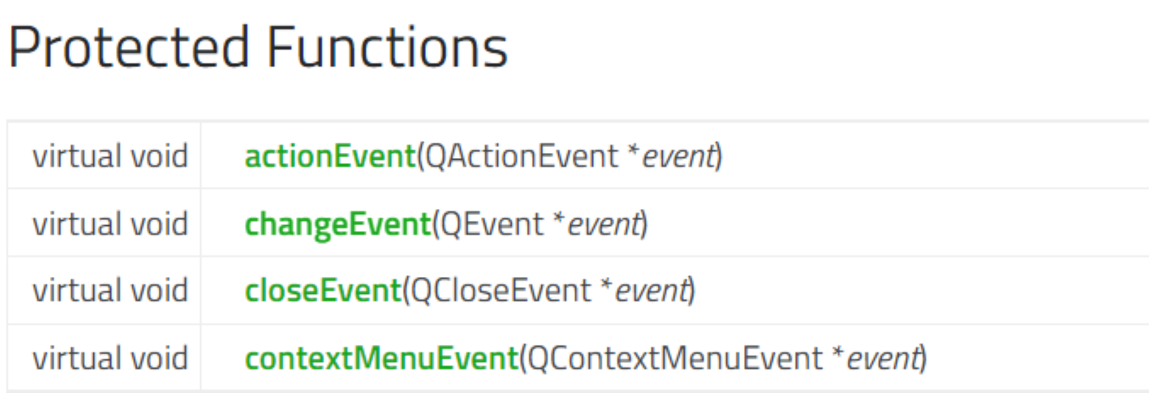
- 事件的处理：
    - 重写虚函数 paintEvent / mousePressEvent / keyPressEvent / …

https://doc.qt.io/qt-6/eventsandfilters.html

事件

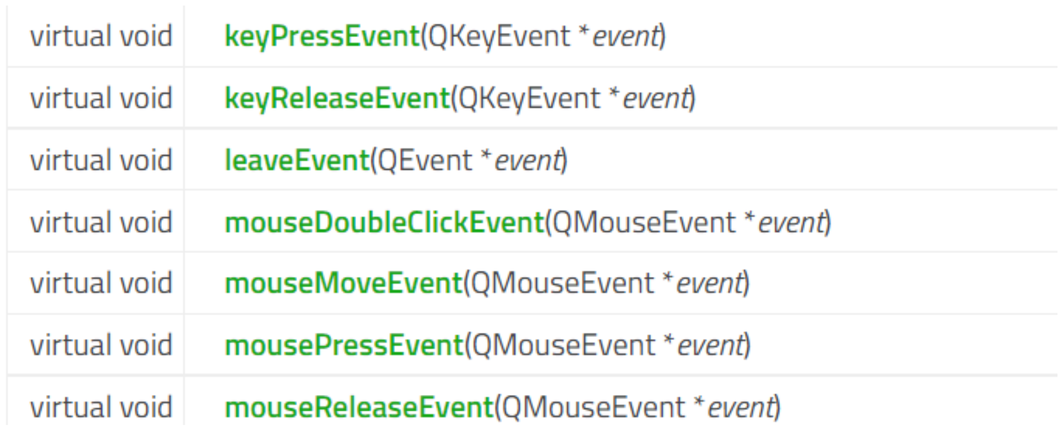
- **事件的处理**
  - 重写虚函数

- **事件的拦截**
  - 事件过滤器
  - 安装：installEventFilter(Qobject*);
  - 移除：removeEventFilter(QObject*);
  - 实现：重写bool eventFilter(QObject*, QEvent*);

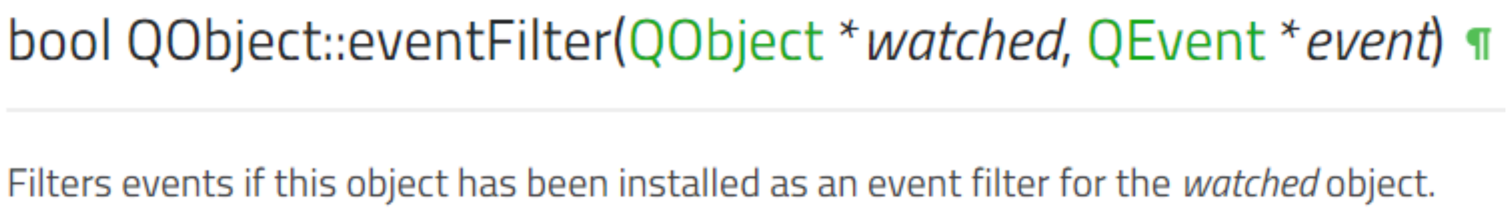


Protected Functions

virtual void	actionEvent(QActionEvent **event*)
virtual void	changeEvent(QEvent **event*)
virtual void	closeEvent(QCloseEvent **event*)
virtual void	contextMenuEvent(QContextMenuEvent **event*)



virtual void	keyPressEvent(QKeyEvent **event*)
virtual void	keyReleaseEvent(QKeyEvent **event*)
virtual void	leaveEvent(QEvent **event*)
virtual void	mouseDoubleClickEvent(QMouseEvent **event*)
virtual void	mouseMoveEvent(QMouseEvent **event*)
virtual void	mousePressEvent(QMouseEvent **event*)
virtual void	mouseReleaseEvent(QMouseEvent **event*)



bool QObject::eventFilter(QObject **watched*, QEvent **event*) ¶

Filters events if this object has been installed as an event filter for the *watched* object.

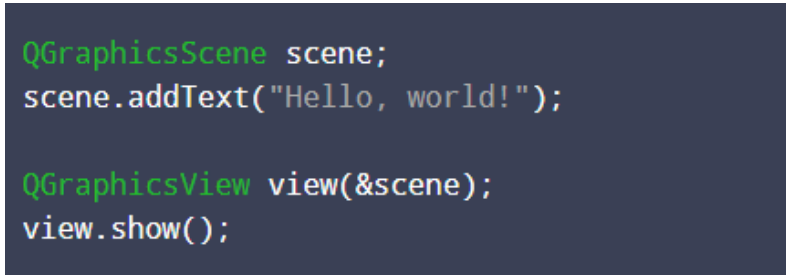
https://doc.qt.io/qt-5.15/qwidget.html

view / scene

视图：可视化场景

QGraphicsView Class

The QGraphicsView class provides a widget for displaying the contents of a QGraphicsScene.



```
QGraphicsScene scene;
scene.addText("Hello, world!");

QGraphicsView view(&scene);
view.show();
```

场景：item的容器

QGraphicsScene Class

The QGraphicsScene class provides a surface for managing a large number of 2D graphical items.

Item: 放置在场景中

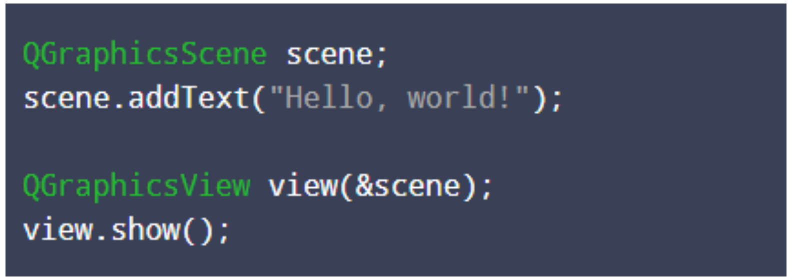
QGraphicsItem Class

The QGraphicsItem class is the base class for all graphical items in a QGraphicsScene.

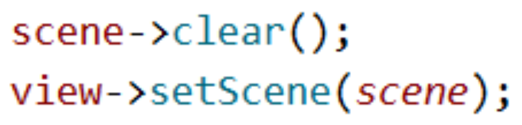
view / scene

场景：

- 添加item: addItem(QGraphicsItem*)
- 移除item: removeItem(QGraphicsItem*)
- 清理：clear()
- 场景改变时发出信号：changed(const QList<QRectF> &*region*)



```
QGraphicsScene scene;
scene.addText("Hello, world!");

QGraphicsView view(&scene);
view.show();
```



```
scene->clear();
view->setScene(scene);
```

Item:

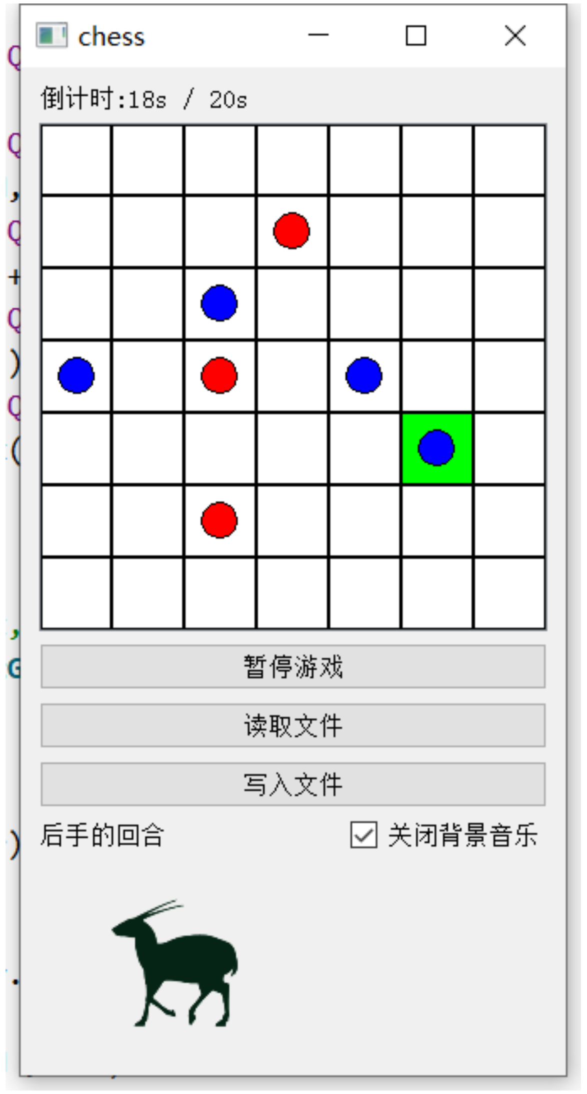
- 设置位置：setPos(qreal, qreal) / setPos(QPointF) / ...

view / scene

继承自QGraphicsRectItem的自定义类：

- 记录位置与格子状态（是否选中，是否由棋子）
- 重写mousePressEvent （下棋）

```
class Grid : public QObject, public QGraphicsRectItem {
  Q_OBJECT
 public:
  Grid(int, int);
  void click();
  int x, y;
  int clicked = -1; // -1: unclicked, 0/1: chess
  void select(bool);

 protected:
  void mousePressEvent(QGraphicsSceneMouseEvent* ) override;

 signals:
  void gridClicked(int, int);
};
```



倒计时:18s / 20s

后手的回合

✓ 关闭背景音乐

view / scene

继承自QGraphicsRectItem的自定义类：

- 重写mousePressEvent（下棋）

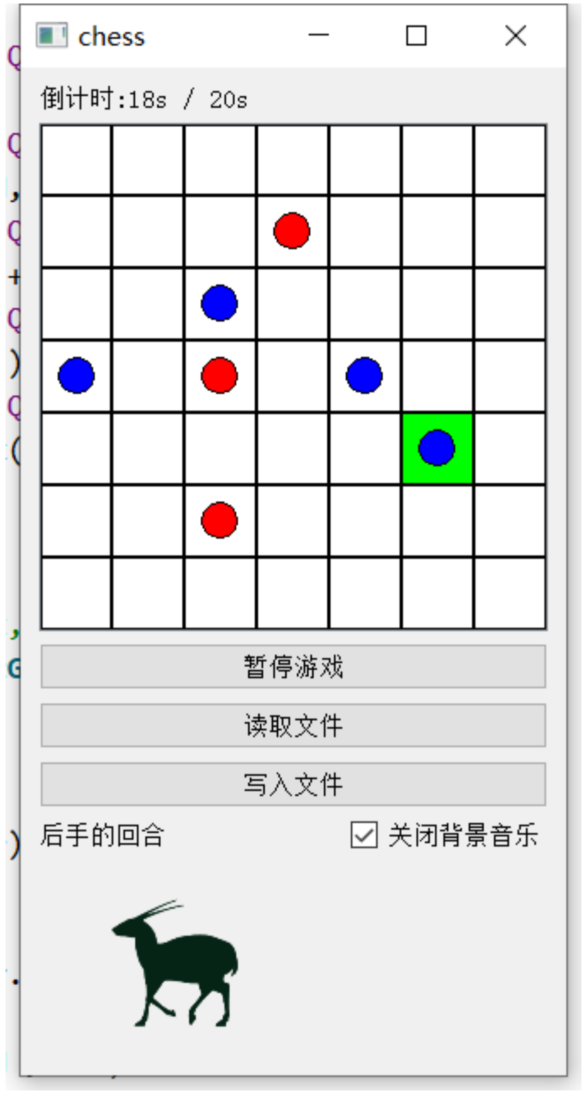
```
void Grid::mousePressEvent(QGraphicsSceneMouseEvent*) {
  click();
}

void Grid::click() {
  if( clicked != -1) return;
  emit gridClicked(x, y);
}

void Grid::select(bool status) {
  if( status ) {
    this->setBrush(QBrush(Qt::green));
  } else {
    this->setBrush(QBrush(Qt::white));
  }
}
```

键盘点击事件

发出gridClicked(x, y)信号

设置格子的选中状态

利用不同的QBrush设置item的颜色



view / scene

在MainWindow中处理：

```
for(int i = 0; i < N ; i++)
  for(int j = 0; j < N; j++) {
    map[i][j] = new Grid(i, j);
    map[i][j]->setRect( (1.0 * i * LENGTH / N), ( 1.0 * j * LENGTH / N),
                        1.0 * LENGTH / N - 1, 1.0 * LENGTH / N - 1);

    connect(map[i][j], SIGNAL(gridClicked(int, int)), this, SLOT(drawChess(int, int)));
    scene->addItem(map[i][j]);
  }
```

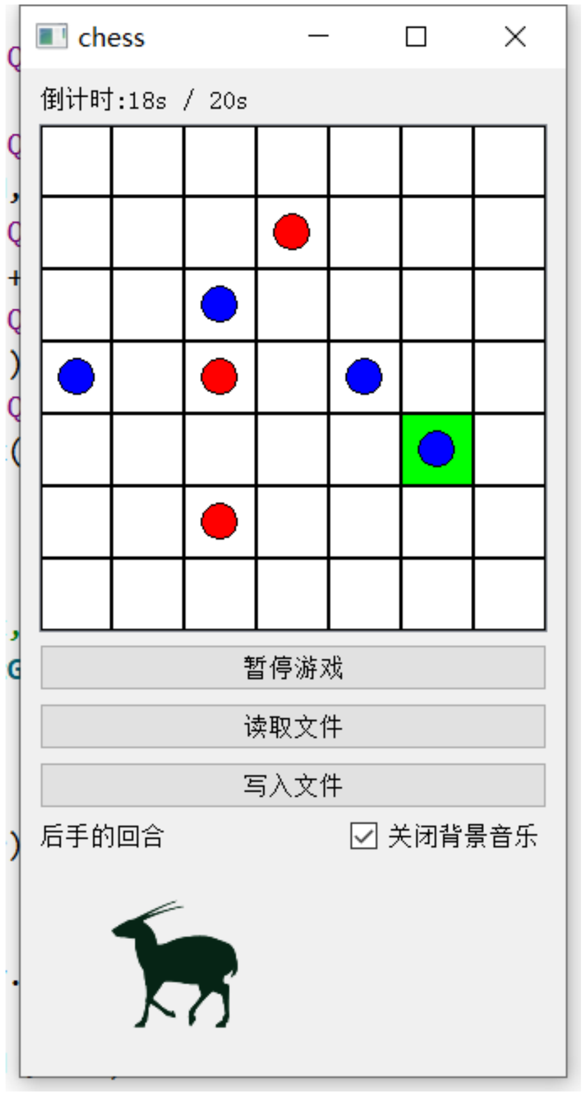
设置item的大小与位置

connect

加入scene

绘制棋子

```
QBrush brush;

if( turn ) brush.setColor(QColor("red"));
else brush.setColor(QColor("blue"));

QGraphicsEllipseItem* chess = new QGraphicsEllipseItem();
chess->setRect( (x * LENGTH / N) + R() / 2.0,   ( y * LENGTH / N) + R() / 2.0, R(), R());
chess->setBrush(brush);
scene->addItem(chess);
```



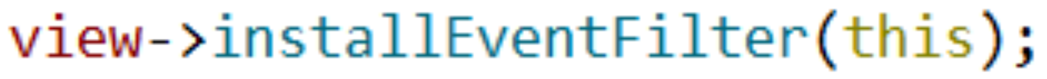
鼠标事件 / 键盘事件 / 事件过滤器

鼠标点击：

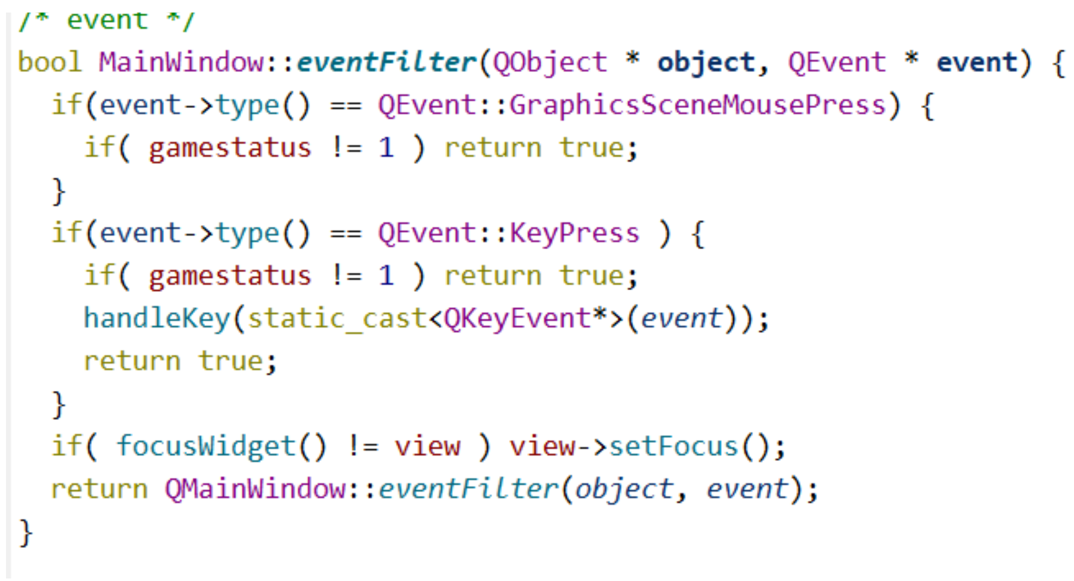
- 重写item的mousePressEvent
- 为view安装事件过滤器
- 在事件过滤器中判断操作是否可行

键盘操作：

- 为view安装事件过滤器
- 设置view为input focus
- 在事件过滤器中判断操作是否可行，并处理键盘事件



```
view->installEventFilter(this);
```



```
/* event */
bool MainWindow::eventFilter(QObject * object, QEvent * event) {
  if(event->type() == QEvent::GraphicsSceneMousePress) {
    if( gamestatus != 1 ) return true;
  }
  if(event->type() == QEvent::KeyPress ) {
    if( gamestatus != 1 ) return true;
    handleKey(static_cast<QKeyEvent*>(event));
    return true;
  }
  if( focusWidget() != view ) view->setFocus();
  return QMainWindow::eventFilter(object, event);
}
```

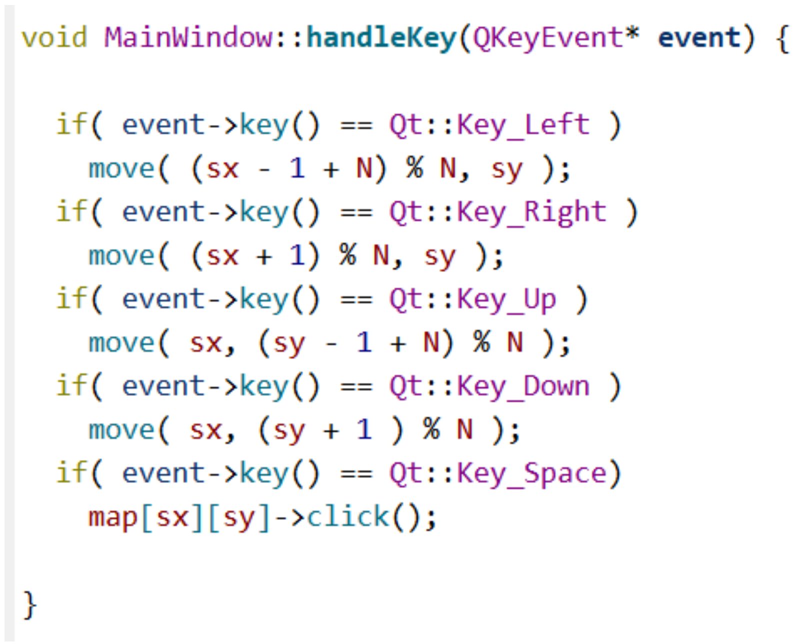
鼠标事件 / 键盘事件 / 事件过滤器

鼠标点击：

- 重写item的mousePressEvent
- 为view安装事件过滤器
- 在事件过滤器中判断操作是否可行

键盘操作：

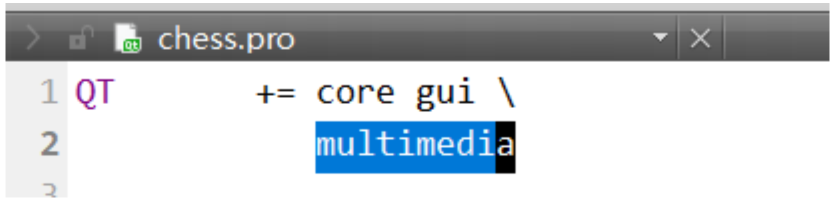
- 为view安装事件过滤器
- 设置view为input focus
- 在事件过滤器中判断操作是否可行，并处理键盘事件



```
void MainWindow::handleKey(QKeyEvent* event) {

  if( event->key() == Qt::Key_Left )
    move( (sx - 1 + N) % N, sy );
  if( event->key() == Qt::Key_Right )
    move( (sx + 1) % N, sy );
  if( event->key() == Qt::Key_Up )
    move( sx, (sy - 1 + N) % N );
  if( event->key() == Qt::Key_Down )
    move( sx, (sy + 1 ) % N );
  if( event->key() == Qt::Key_Space)
    map[sx][sy]->click();

}
```

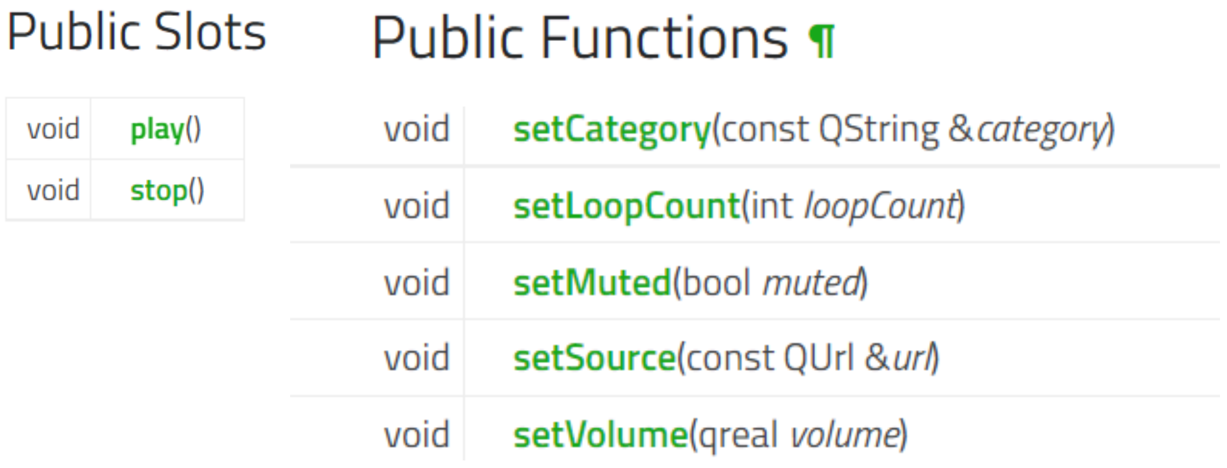
音乐 / 音效

- **准备工作**
- QSoundEffect
- QMeidaPlayer



chess.pro

```
QT       += core gui \
            multimedia
```



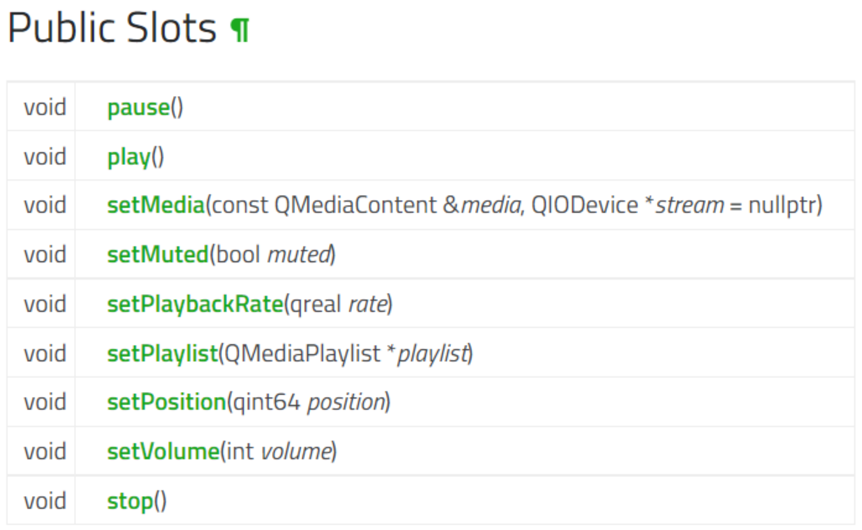
Public Slots

void	play()
void	stop()

Public Functions ¶

void	setCategory(const QString &*category*)
void	setLoopCount(int *loopCount*)
void	setMuted(bool *muted*)
void	setSource(const QUrl &*url*)
void	setVolume(qreal *volume*)

QSoundEffect



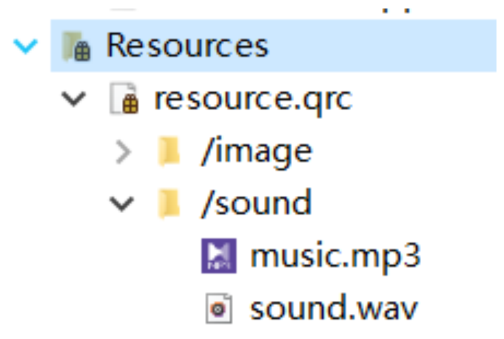
Public Slots ¶

void	pause()
void	play()
void	setMedia(const QMediaContent &*media*, QIODevice **stream* = nullptr)
void	setMuted(bool *muted*)
void	setPlaybackRate(qreal *rate*)
void	setPlaylist(QMediaPlaylist **playlist*)
void	setPosition(qint64 *position*)
void	setVolume(int *volume*)
void	stop()

QMediaPlayer

音乐 / 音效

- **准备工作**
- QSoundEffect
- QMeidaPlayer



```
Resources
    resource.qrc
        /image
        /sound
            music.mp3
            sound.wav
```

```
QSoundEffect* sound; // sound effect
QMediaPlayer* player; // bgm music player
```

```
sound = new QSoundEffect();
sound->setSource(QUrl("qrc:/sound/sound.wav"));
sound->setVolume(0.2);
```

路径设置

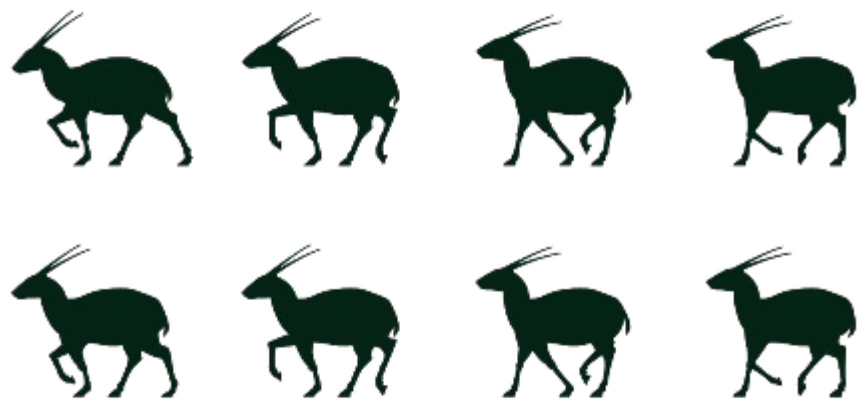
```
/* sound */
sound->play();
```

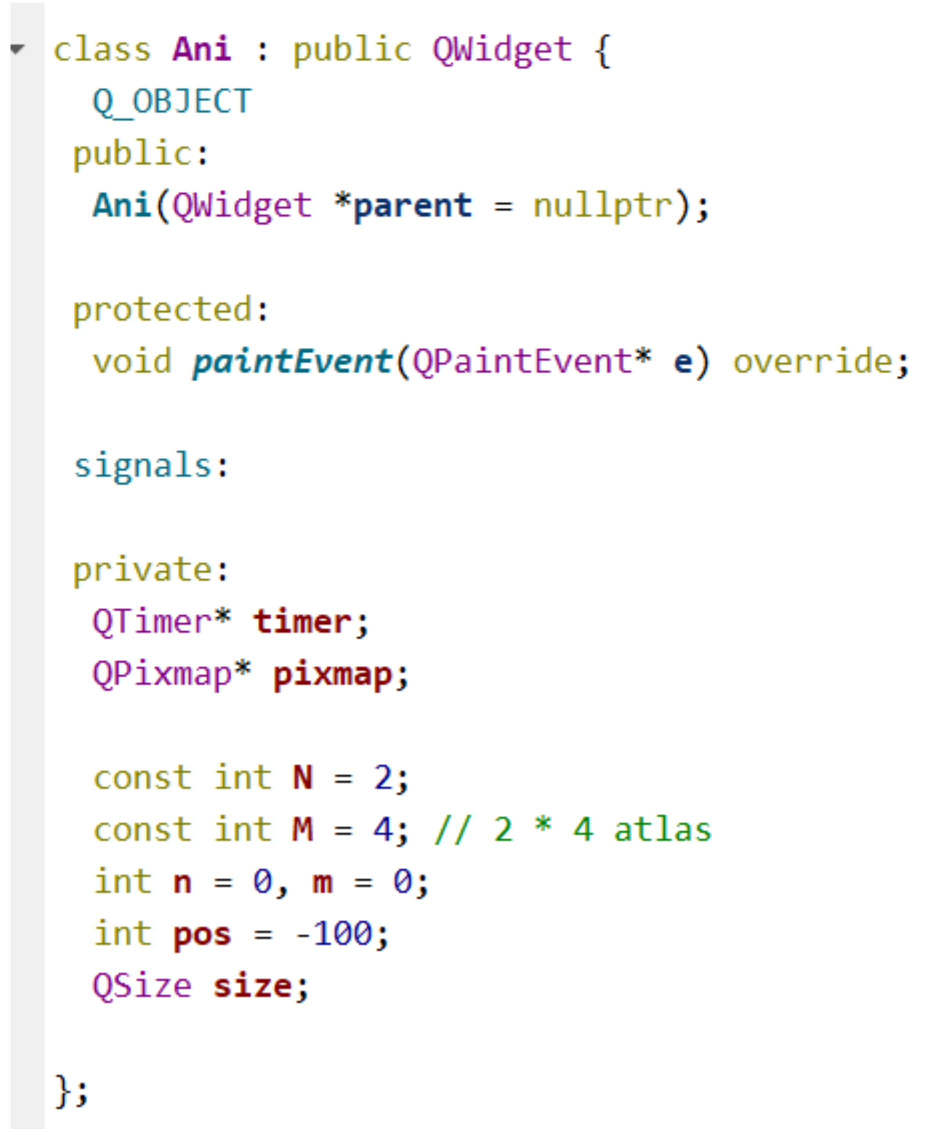
```
player = new QMediaPlayer();
player->setMedia(QUrl("qrc:/sound/music.mp3"));
player->setVolume(10);
```

```
player->play();        player->pause();
player->setMuted(status != Qt::Unchecked);
```

简易动画

- 由多个**帧**连续快速播放的简单动画
- **图集** (atlas)
- 自定义继承自QWidget的类以完成动画
    - 加载图片资源 (QPixmap)
    - 定时切换帧 (QTimer)
    - 播放对应帧 (update / paintEvent)

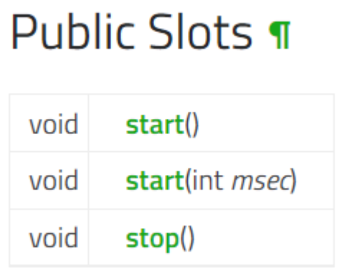




```cpp
class Ani : public QWidget {
  Q_OBJECT
 public:
  Ani(QWidget *parent = nullptr);

 protected:
  void paintEvent(QPaintEvent* e) override;

 signals:

 private:
  QTimer* timer;
  QPixmap* pixmap;

  const int N = 2;
  const int M = 4; // 2 * 4 atlas
  int n = 0, m = 0;
  int pos = -100;
  QSize size;

};
```

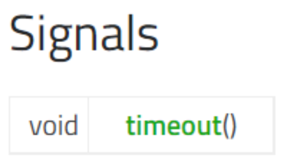
简易动画

QTimer定时器



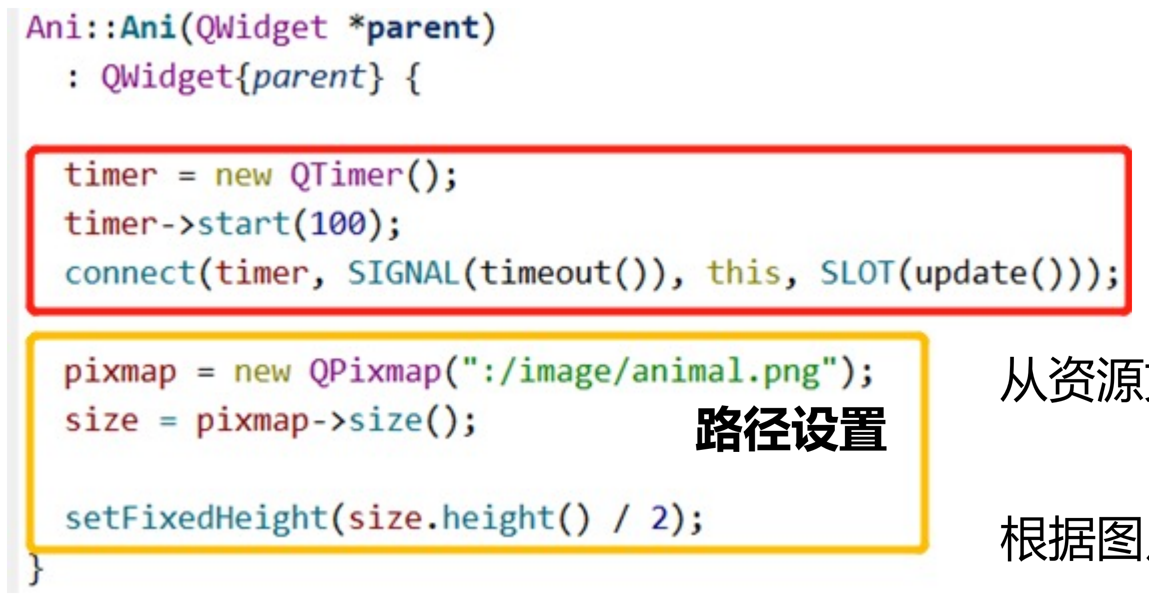
Public Slots ¶

void	start()
void	start(int *msec*)
void	stop()



Signals

void	timeout()

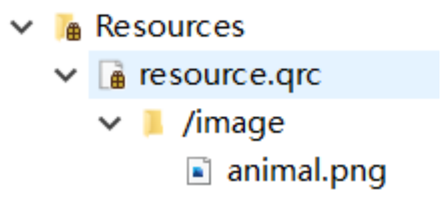


```
Ani::Ani(QWidget *parent)
  : QWidget{parent} {

  timer = new QTimer();
  timer->start(100);
  connect(timer, SIGNAL(timeout()), this, SLOT(update()));

  pixmap = new QPixmap(":/image/animal.png");
  size = pixmap->size();

  setFixedHeight(size.height() / 2);
}
```

路径设置

设置100ms的定时器，与update槽关联

从资源文件中加载图片资源



Resources
resource.qrc
/image
animal.png

根据图片高度确定widget高度

Tips:

- 使用Qt的资源文件
- 善用QPixmap的isNull等方法确定问题所在

简易动画

- 重写paintEvent
- 在不同帧绘制pixmap的不同部分

```
Ani::Ani(QWidget *parent)
  : QWidget{parent} {

  timer = new QTimer();
  timer->start(100);
  connect(timer, SIGNAL(timeout()), this, SLOT(update()));

  pixmap = new QPixmap(":/image/animal.png");
  size = pixmap->size();

  setFixedHeight(size.height() / 2);
}
```

void QWidget::paintEvent(QPaintEvent *event) ¶ [virtual protected]

A paint event is a request to repaint all or part of a widget. It can happen for one of the following reasons:

› repaint() or update() was invoked,

```
protected:
  void paintEvent(QPaintEvent* e) override;
```

```
void Ani::paintEvent(QPaintEvent*) {
  QPainter painter(this);
  m += 1;
  if( m == M ) {
    n = (n + 1) % N;
    m = 0;
  }

  pos -= 10;
  if( pos < -10 ) pos = width() + 10;

  painter.drawPixmap(pos - 100, 0, *pixmap,
                  m * size.width() / M, n * size.height() / N, size.width() / M, size.height() / N);
}
```

计算当前帧的位置

计算pixmap应在widget的横坐标何处显示

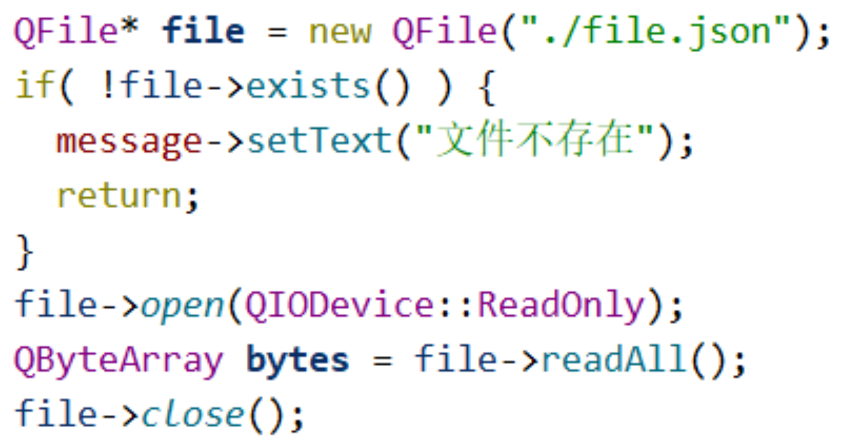
pixmap绘制在widget的位置

绘制pixmap区域的左上角

绘制pixmap区域的大小

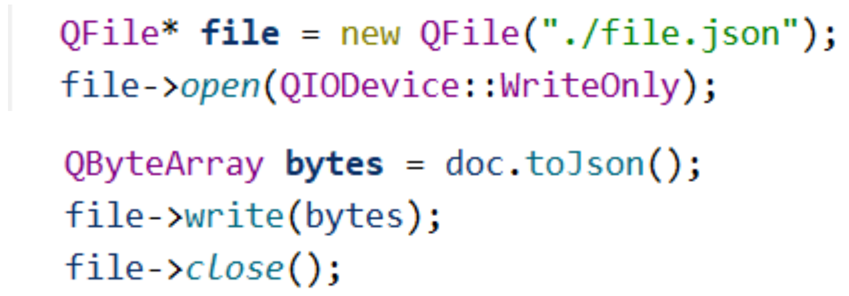
文件读取

- 文件的读取与写入 （ QFile <-> QByteArray ）
- Json的解析与生成 (QByteArray <-> QJsonDocument <-> ...)



```
QFile* file = new QFile("./file.json");
if( !file->exists() ) {
  message->setText("文件不存在");
  return;
}
file->open(QIODevice::ReadOnly);
QByteArray bytes = file->readAll();
file->close();
```

文件的读取

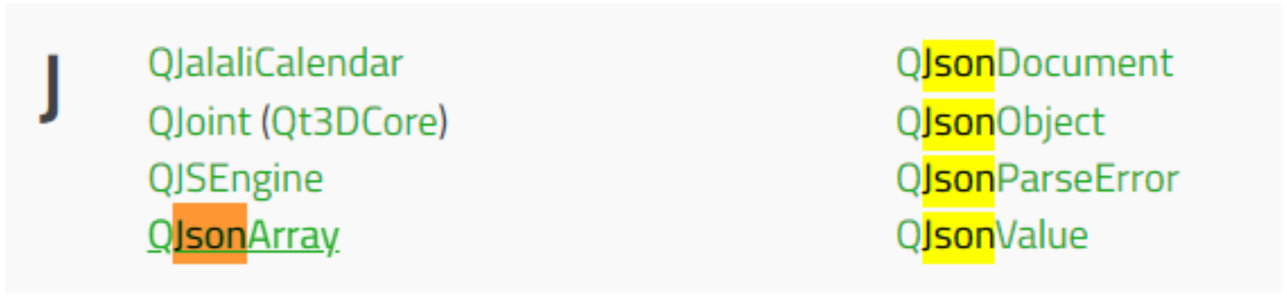


```
QFile* file = new QFile("./file.json");
file->open(QIODevice::WriteOnly);

QByteArray bytes = doc.toJson();
file->write(bytes);
file->close();
```

文件的写入

文件读取

- Json的解析与生成



J	QJalaliCalendar	QJsonDocument
	QJoint (Qt3DCore)	QJsonObject
	QJSEngine	QJsonParseError
	QJsonArray	QJsonValue

QJsonDocument Class

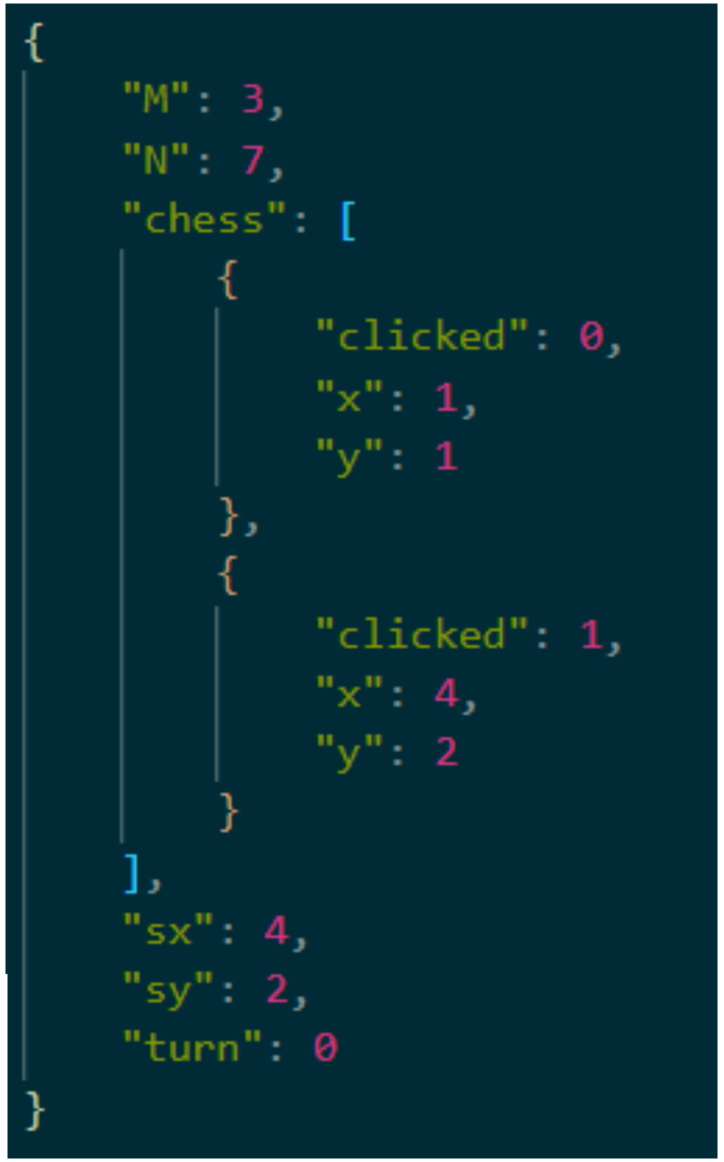
The QJsonDocument class provides a way to read and write JSON documents. More...

QJsonArray Class

The QJsonArray class encapsulates a JSON array.

QJsonObject Class

The QJsonObject class encapsulates a JSON object.



```
{
    "M": 3,
    "N": 7,
    "chess": [
        {
            "clicked": 0,
            "x": 1,
            "y": 1
        },
        {
            "clicked": 1,
            "x": 4,
            "y": 2
        }
    ],
    "sx": 4,
    "sy": 2,
    "turn": 0
}
```

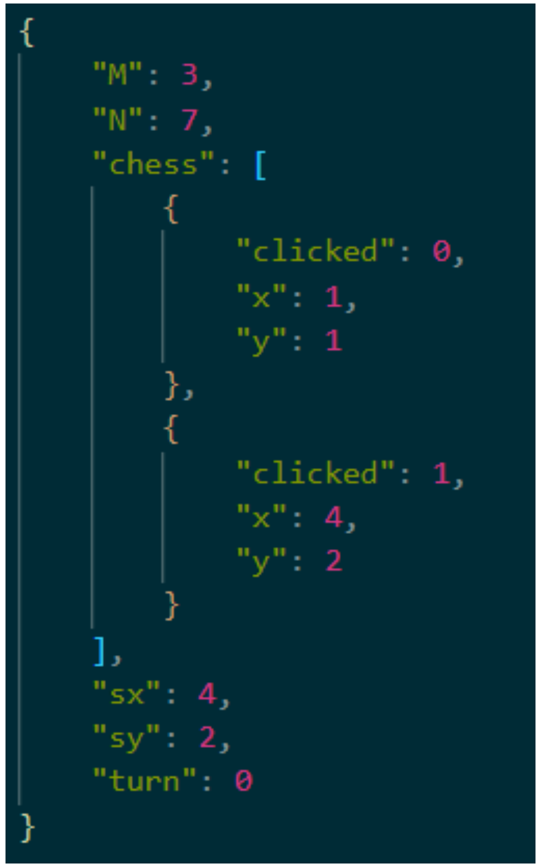
文件读取

- Json的解析（QByteArray -> QJsonDocument -> int/string/...)

```
QJsonDocument doc = QJsonDocument::fromJson(bytes);
QJsonObject info = doc.object();
N = info.value("N").toInt();
M = info.value("M").toInt();
QJsonArray array = info.value("chess").toArray();

startGame();

foreach(QJsonValue v, array) {
  QJsonObject obj = v.toObject();
  int x = obj.value("x").toInt();
  int y = obj.value("y").toInt();
  int clicked = obj.value("clicked").toInt();
  putChess(x, y, clicked);
}

turn = info.value("turn").toInt();
```

读取对应value并转化为int

遍历QJsonArray



```
{
    "M": 3,
    "N": 7,
    "chess": [
        {
            "clicked": 0,
            "x": 1,
            "y": 1
        },
        {
            "clicked": 1,
            "x": 4,
            "y": 2
        }
    ],
    "sx": 4,
    "sy": 2,
    "turn": 0
}
```

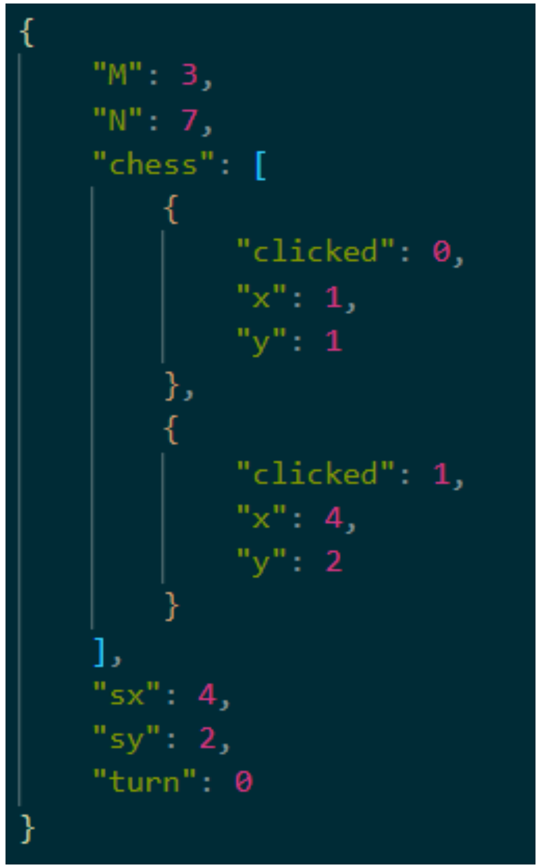
文件读取

- Json的生成（QByteArray <- QJsonDocument <- object/array...)

```
QJsonArray array;
for(auto& v : map)
  for(auto& g : v) {
    if( g->clicked != -1 ) {
      QJsonObject o;
      o.insert("x", g->x);
      o.insert("y", g->y);
      o.insert("clicked", g->clicked);
      array.append(o);
    }
  }
QJsonObject info;
info.insert("chess", array);
info.insert("N", N);
info.insert("M", M);
info.insert("turn", turn);
info.insert("sx", sx);
info.insert("sy", sy);
QJsonDocument doc(info);
QByteArray bytes = doc.toJson();
```

构造array

构造其余信息



```
{
    "M": 3,
    "N": 7,
    "chess": [
        {
            "clicked": 0,
            "x": 1,
            "y": 1
        },
        {
            "clicked": 1,
            "x": 4,
            "y": 2
        }
    ],
    "sx": 4,
    "sy": 2,
    "turn": 0
}
```

windeployqt

编译指令

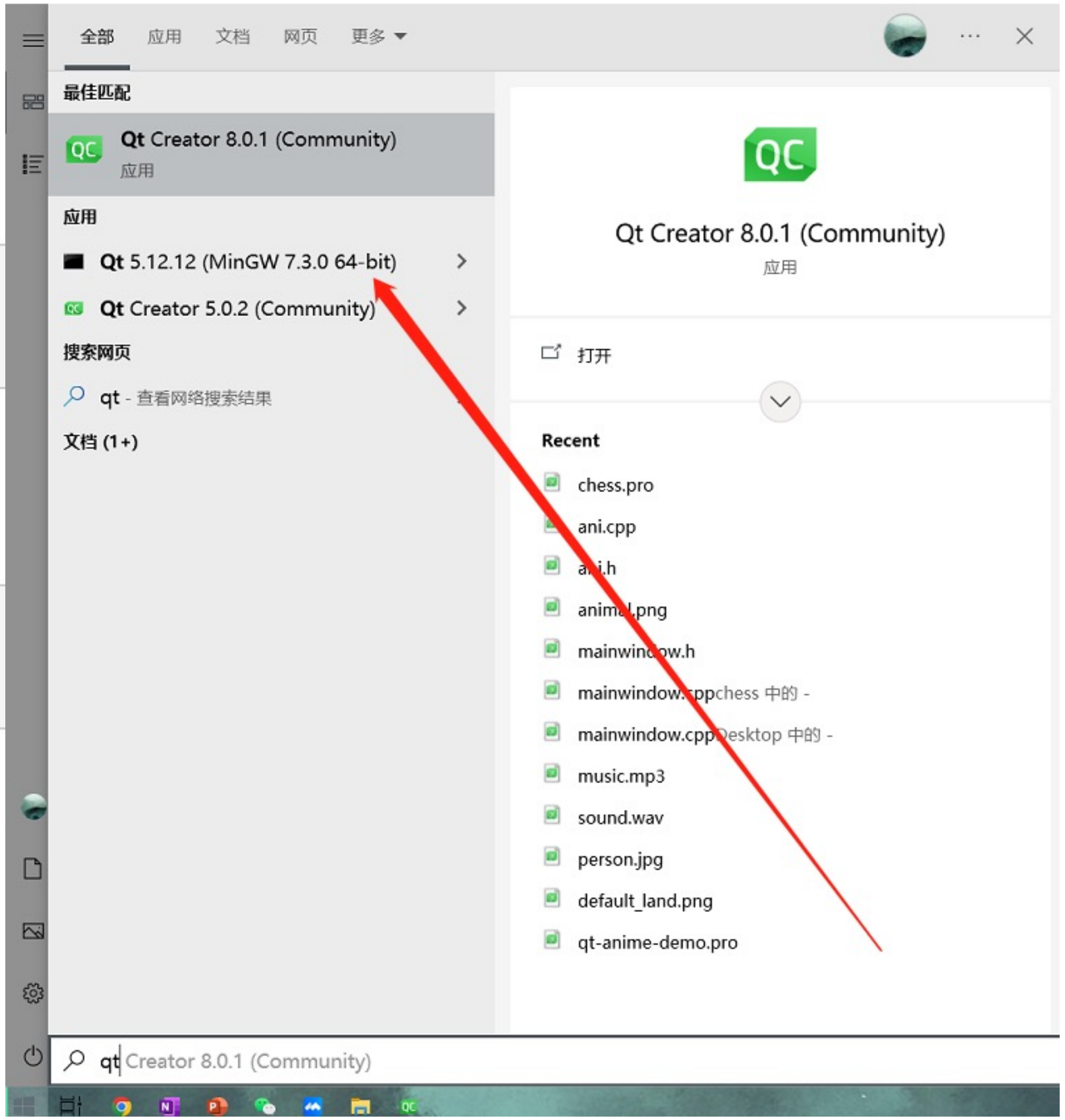
```
qmake
```

程序打包

使用Qt自带的 MinGW，在其中运行打包程序

```
windeployqt homework4-1.exe
```

打包成功后会自动添加缺失的动态库`.dll`文件，可以直接运行。



全部
应用
文档
网页
更多
最佳匹配
Qt Creator 8.0.1 (Community)
应用
应用
Qt 5.12.12 (MinGW 7.3.0 64-bit)
Qt Creator 5.0.2 (Community)
搜索网页
qt - 查看网络搜索结果
文档 (1+)
Qt Creator 8.0.1 (Community)
应用
打开
Recent
chess.pro
ani.cpp
ani.h
animal.png
mainwindow.h
mainwindow.cppchess 中的 -
mainwindow.cppDesktop 中的 -
music.mp3
sound.wav
person.jpg
default_land.png
qt-anime-demo.pro
qt Creator 8.0.1 (Community)

Qt基本功能

2022.08.24

https://github.com/glassesq/chess