СЕМИНАР 7.2

Механизм событий Qt
Чем отличается от сигналов и слотов?
События QEvent
Обработчики событий
Фильтры событий
События и рисование
События клавиатуры, мыши и т.п.

Механизм событий Qt

Qt – event driven Ui toolkit
Народная мудрость

Событие – объект класса QEvent, который предоставляет информацию об изменениях, которые произошли в или вне (события Window System, ОС и т.п.) приложения.

События принимаются и обрабатываются классами - наследниками QObject.

Примеры событий:

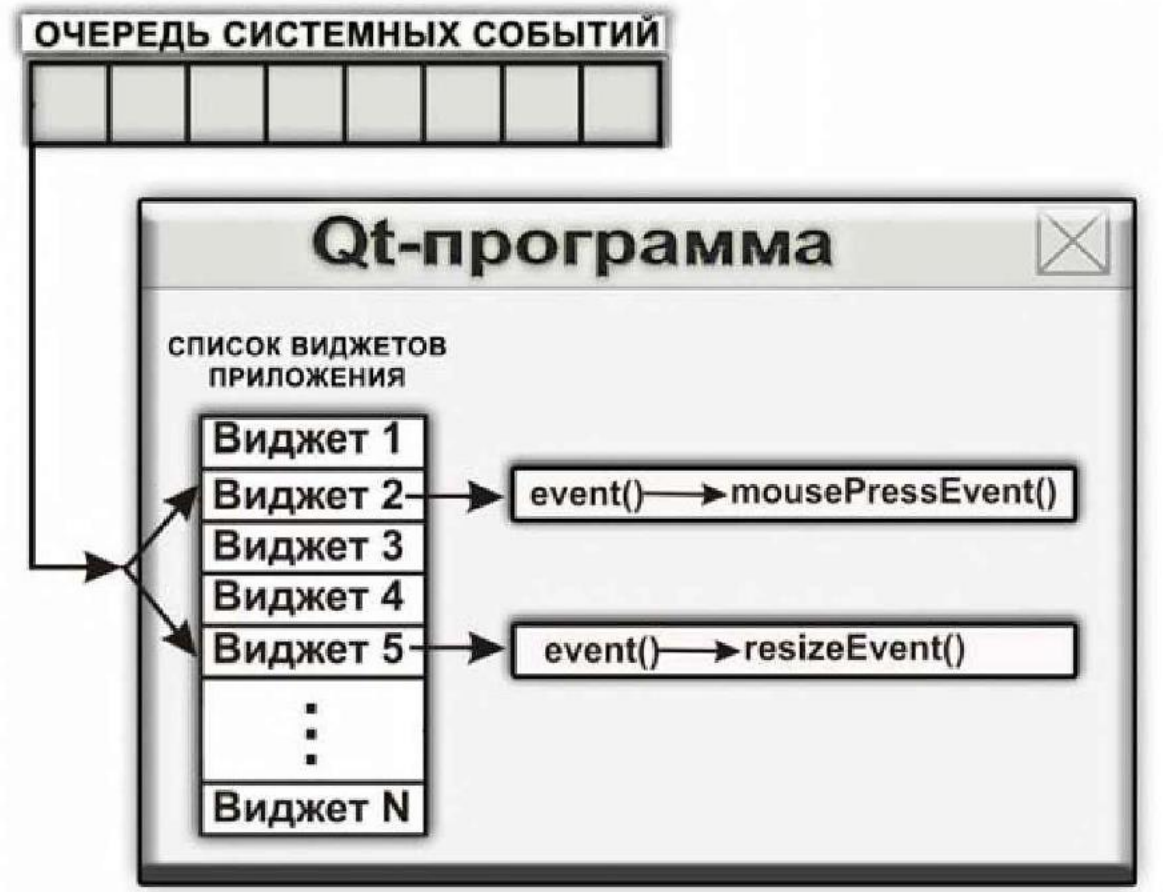
- Нажатие кнопки клавиатуры;
- Щелчок кнопки мыши;
- Перемещение, изменение окна приложения;
- Тик таймера;
- Нажатие кнопки закрытия окна приложения;
- И т.д...

Чем отличается от сигналов и слотов?

1. События обрабатываются одним методом
2. У каждого события есть получатель (наследник QObject);
3. Могут быть приняты или проигнорированы;
    1. event->accept(); event->ignore();
4. События обрабатываются в цикле обработки событий:
    1. QCoreApplication::exec();
    2. QApplication::exec();
    3. QCoreApplication::processEvents();
5. Представляют собой более низкий уровень программной реализации
    1. Используются при создании некоторых типов сигналов:
    2. Например сигнал clicked() выпускается из метода обработки событий мыши mousePressEvent();

Механизм событий

Механизм событий



Источники событий

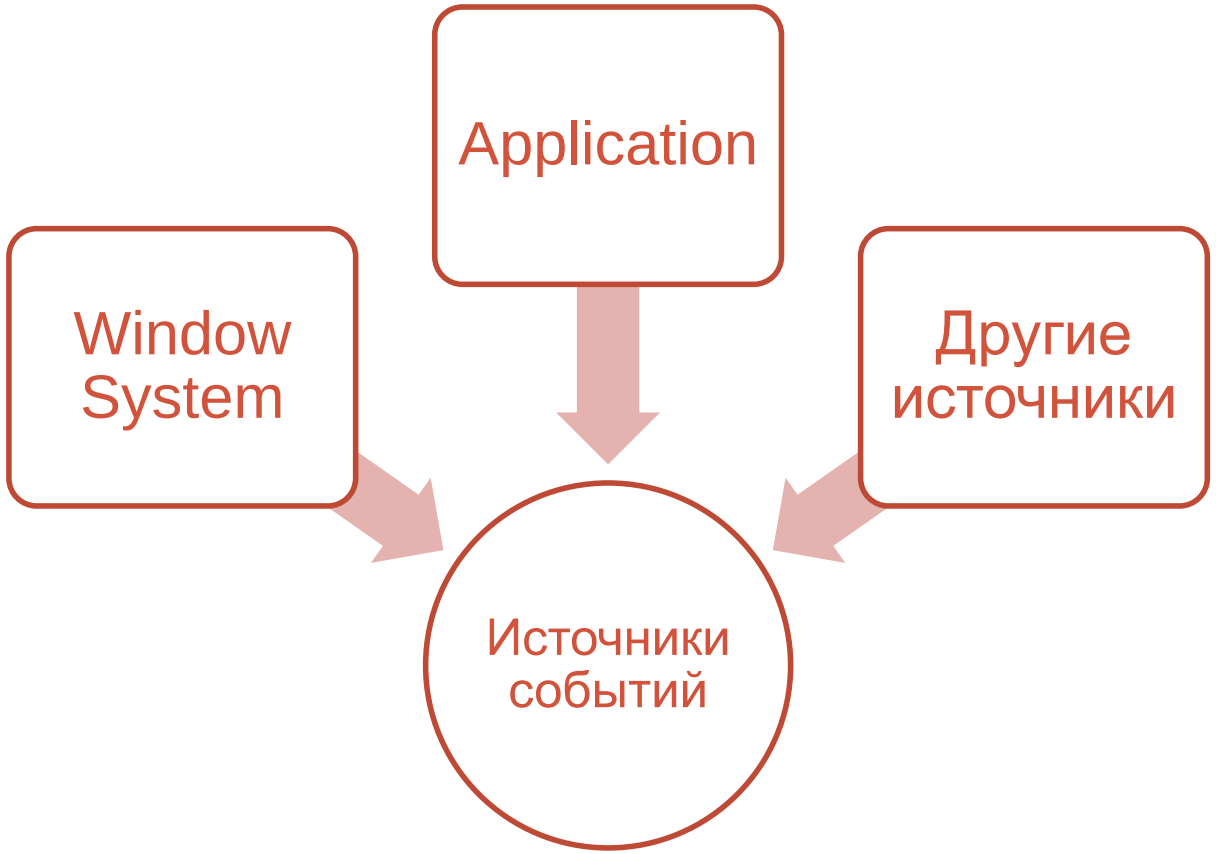
- 1. *Spontaneous events* – события, созданные window system. Обрабатываются в event loop в exec()
- 2. *Posted events – события, вызванные из Qt или приложения. Обрабатываются в* event loop в exec();
- 3. *Sent events* – события, вызванные из Qt или приложения. Отправляются непосредственно целевому объекту. Не обрабатываются в event loop в exec().

Событие и QEvent

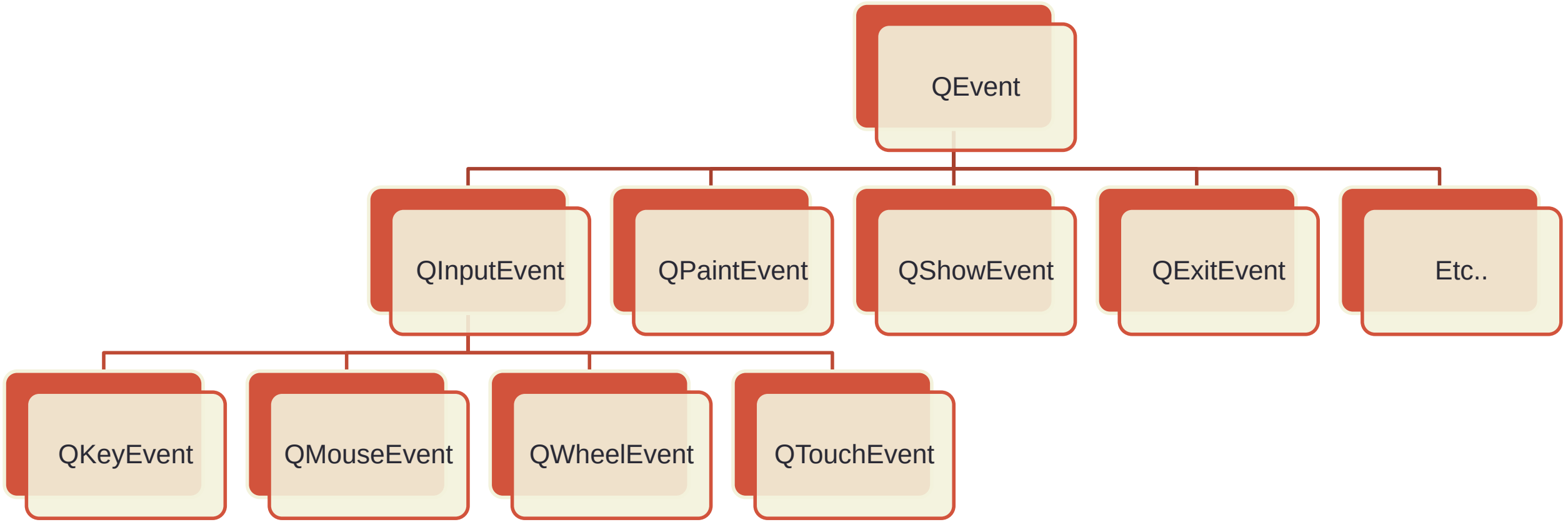
- QEvent – базовый класс событий.

Событие имеет следующие атрибуты:

1. Тип – ID идентификатор события (существует для каждого события, устанавливается в конструкторе);
2. Полезные данные (координаты курсора и т.п.);
3. Флаг о том принято событие или проигнорировано.



Событие и QEvent

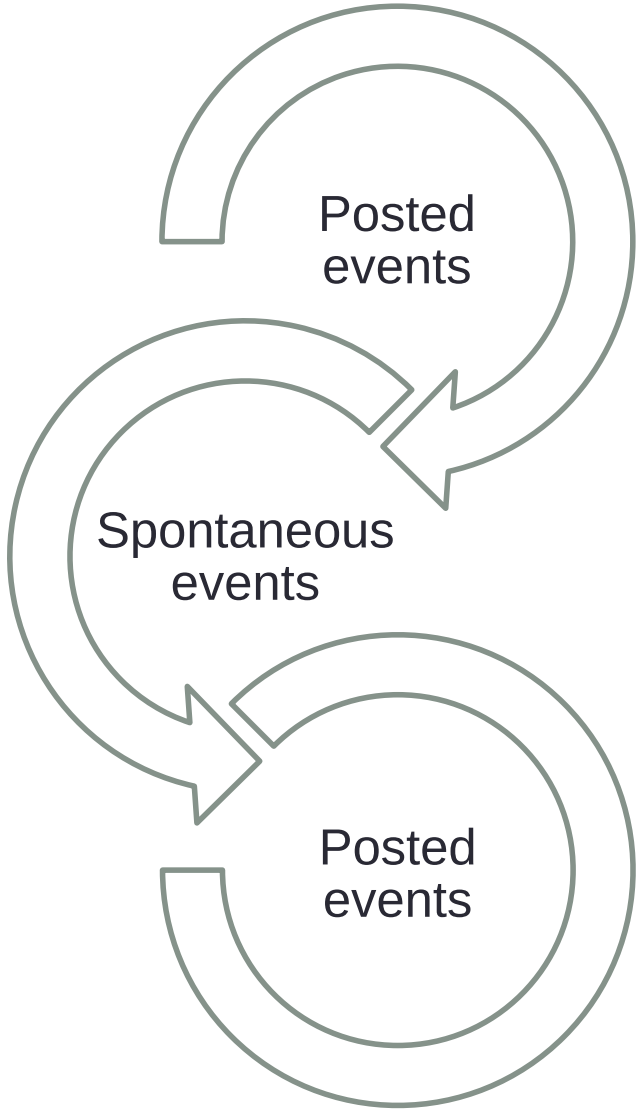


Обработка очереди событий

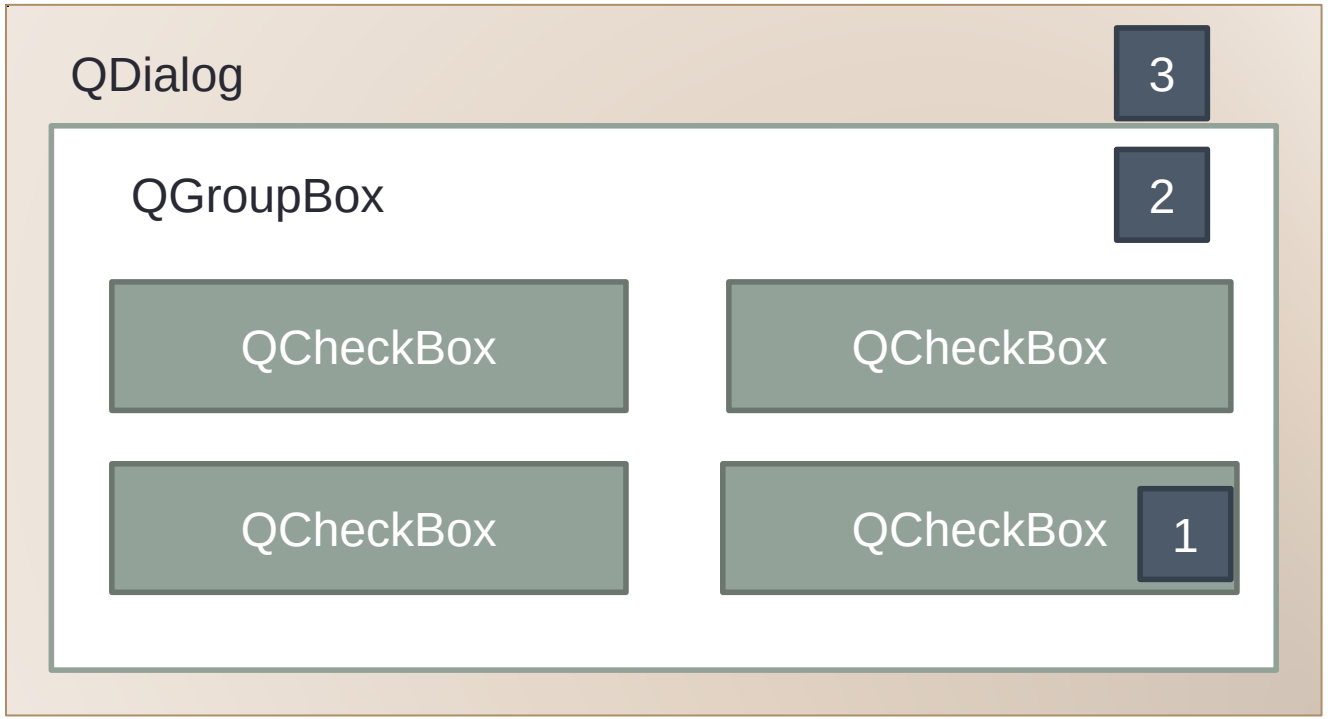
- Функция Qapplication::exec()

```
while (!exit_was_called) {
        while (!posted_event_queue_is_empty) {
                process_next_posted_event();
        }
        while (!spontaneous_event_queue_is_empty) {

        process_next_spontaneous_event();
        }
        while (!posted_event_queue_is_empty) {
                process_next_posted_event();
        }
}
```



Доставка события



- Порядок обработки события:
- 1. Виджет, который находится в фокусе.
- 2. родительский виджет
- 3. и т. п.

Механизм обработки событий

QApplication::notify()

- Доставляет событие объекту
- Возвращает принят объект или нет
- Если объект не принял события, то передает их родителю

Фильтр событий QApplication

Фильтр событий объектов

Метод event() объекта

eventHandler() для данного типа событий

его типа передает его на обработку тому

Фильтр событий

Фильтры событий позволяют перехватить событие раньше объекта, которому предназначалось событие.

Для того, чтобы реализовать фильтр событий необходимо:

- Класс – наследник QObject;
- В этом классе должен быть переопределен метод

  bool eventFilter(QObject* receiver, QEvent *event)

  receiver – указатель на объект, которому предназначалось событие

  event – указатель на событие

  метод возвращает true – событие не передается дальше

  false – событие передается дальше
- Для объекта, события которого перехватываются фильтром вызвать метод:
- installEventFilter (указатель на объект с фильтром)

Фильтр событий

```
KeyPressEater *keyPressEater = new KeyPressEater(this);
QPushButton *pushButton = new QPushButton(this);
QListView *listView = new QListView(this);

pushButton->installEventFilter(keyPressEater);
listView->installEventFilter(keyPressEater);
```

```
bool KeyPressEater::eventFilter(QObject *obj, QEvent *event)
{
    if (event->type() == QEvent::KeyPress) {
        QKeyEvent *keyEvent = static_cast<QKeyEvent *>(event);
        qDebug("Ate key press %d", keyEvent->key());
        return true;
    } else {
        // standard event processing
        return QObject::eventFilter(obj, event);
    }
}
```

Способы обработки и фильтрации событий

QApplication::notify()	• Создать свой класс – наследник QApplication и переопределить в нем метод notify() • Едиственный способ получить доступ ко всем событиям до фильтров
Фильтр событий Qapp	• Создать объект фильтра • Выполнить метод installEventFilter() для приложения
Фильтр событий объектов	• Создать объект фильтра • Выполнить метод installEventFilter() для объектов
Метод event() объекта	• Переопределить метод event() для объекта
eventHandler() для данного типа событий	• Переопределить eventHandler (Например метод paintEvent, timerEvent, closeEvent, keyPressedEvent и т.п.)

Переопределение метода event()

```
bool QWidget::event(QEvent *event)
{
    switch (e->type()) {
    case QEvent::KeyPress:
        keyPressEvent((QKeyEvent *)event);
        if (!((QKeyEvent *)event)->isAccepted())
            return false;
        break;
    case QEvent::KeyRelease:
        keyReleaseEvent((QKeyEvent *)event);
        if (!((QKeyEvent *)event)->isAccepted())
            return false;
        break;
        ...
    }
    return true;
}
```

Переопределение eventHandler’a

```
void PicFrame::keyPressEvent(QKeyEvent *e)
{

    switch (e->key()) {
    case Qt::Key_D:
        yaw++;
        qDebug()<<yaw;
        this->update();
        break;
    case Qt::Key_A:
        yaw--;
        qDebug()<<yaw;
        this->update();
        break;
    default:
        QFrame::keyPressEvent(e);
        break;
    }
}
```

```
void PicFrame::mousePressEvent(QMouseEvent *e)
{

    float x= e->localPos().x()-this->width()/2;
    float y= e->localPos().y()-this->height()/2;
    if (x!=0) yawDesirable= 90+atan(y/x)*57.3;
    update();
}
```

```
void CompassForm::closeEvent(QCloseEvent *event)
{
    if (windowShouldClose) event->accept();
    else event->ignore();
}
```

Практическая часть

- Дополнить проект следующим функционалом:
- Заданное положение курса (положение красой внешней стрелки) должно соответствовать положению курсора мыши на экране
- Изменение значения заданного и текущего курса по нажатию кнопок
  - Должно изменяться положение стрелки и цифровых индикаторов
- Если включен флаг Close Mode, то приложение нельзя закрыть.