Семинар «Технологии QNX и КПДА в России»

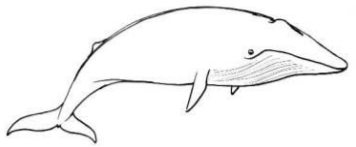
Санкт-Петербург, 30 октября 2018

Проектирование приложений с интерфейсами HID и Multitouch на основе фреймворка Qt под ЗОСРВ «Нейтрино»

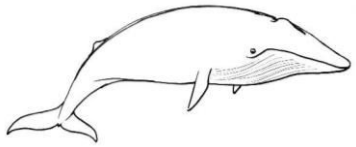
Николай Коберда, инженер-программист
ООО «СВД Встраиваемые Системы»

СВД Встраиваемые Системы

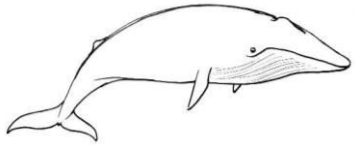
SWD Software



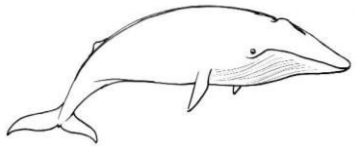
Драйвер системы ввода-вывода HID



Графическая подсистема Photon

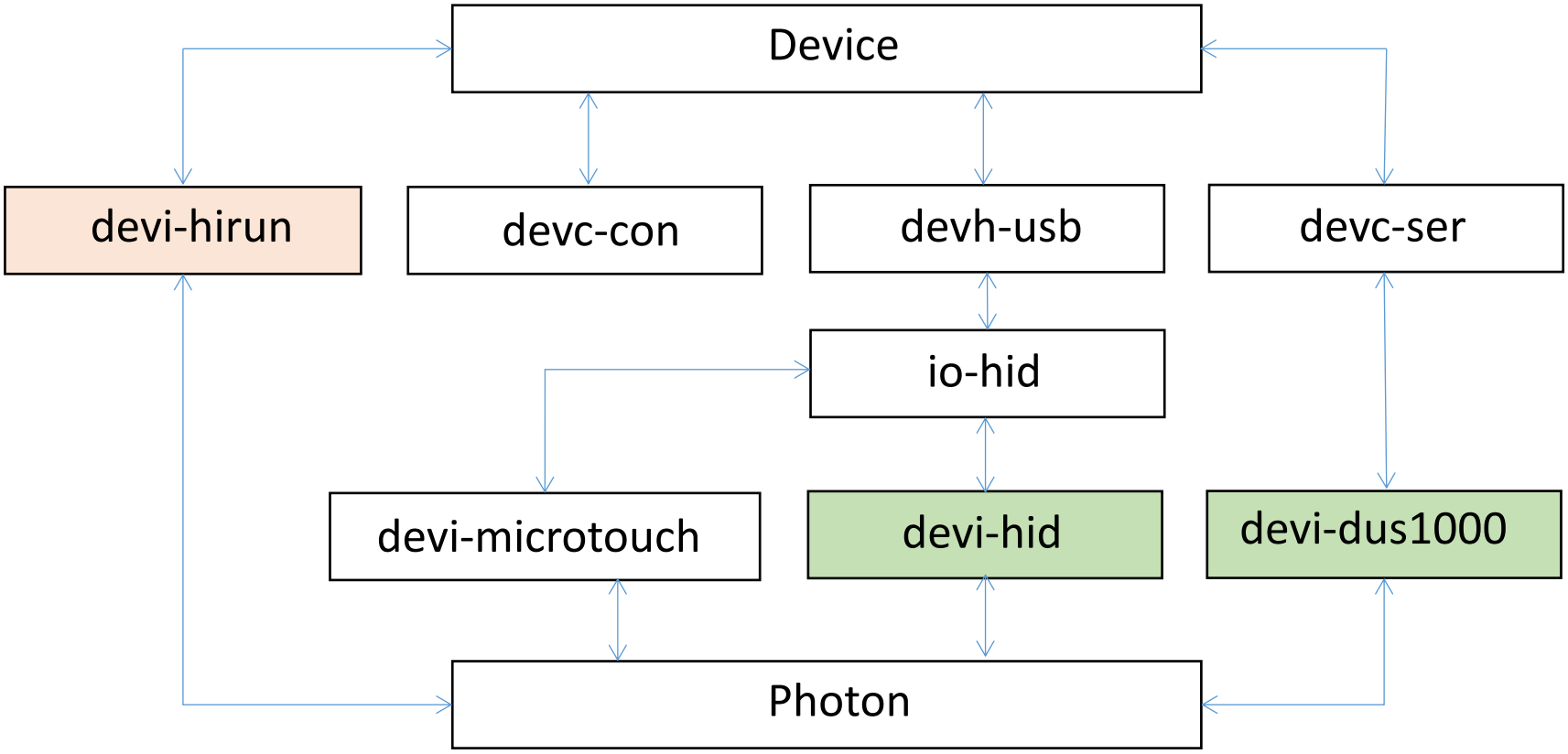


Библиотека виджетов libph.so



Плагины и библиотеки Qt

Ключевые компоненты подсистемы поддержки HID устройств



Реализована поддержка multitouch

Не рекомендуется к использованию одновременно с io-hid

Варианты поддерживаемых устройств:

Решаемые драйвером задачи:

- детектирование режима функционирования HID-устройства;
- обслуживание инициализации устройства и перехват входящего трафика;
- определение формата приходящих пакетов данных;
- передача данных в оконное окружение Photon в виде стандартизованных событий (Events).

Передача событий ввода в оконное окружение Photon

Основная задача:

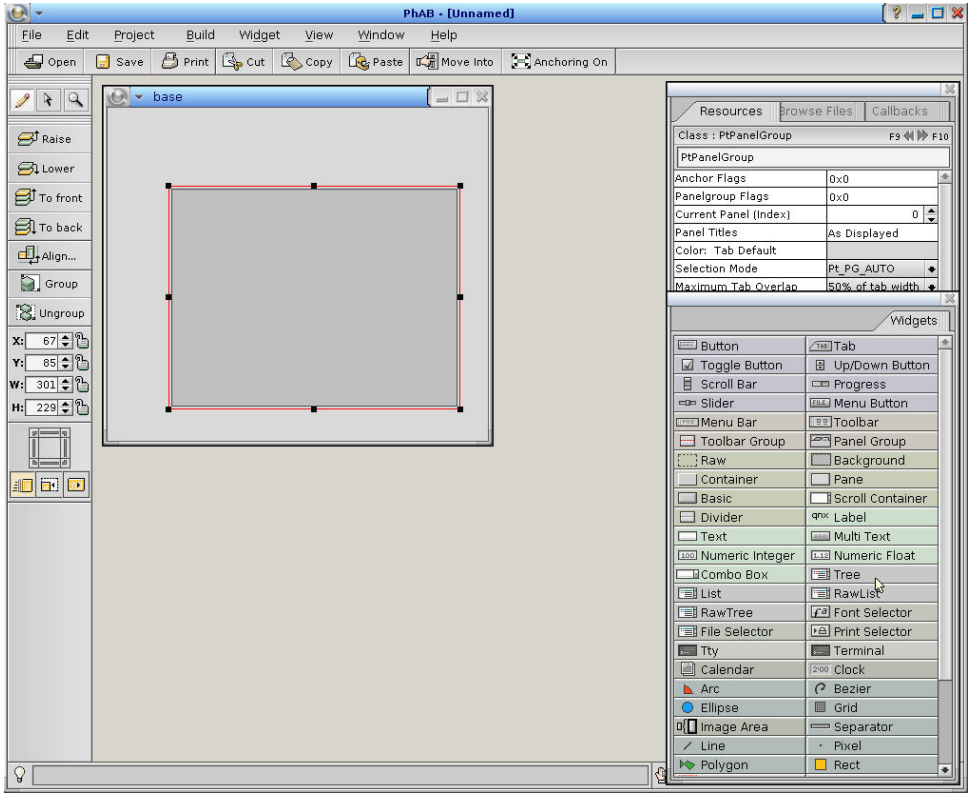
- анализ пришедшего от HID-драйвера события (пересчет координат, привязка к дисплею и окну приложения, …);
- определение типа события (Drag-and-Drop, Press, Unpress);
- создание события для последующей отправки в системную библиотеку (libph.so).

Структура события для позиционируемого устройства ввода:

- тип (press, unpress, steady);
- подтип (real, phantom, …);
- координаты событий (абсолютные, относительные);
- состояние кнопки;
- различные флаги (z-index, …);
- метка наличия нескольких источников данных (**ключевое поле для multitouch**);
- метка времени.

Библиотека предоставляет обширное API, включая штатные оконные шаблоны и ключевые примитивы:

- окна;
- кнопки;
- списки;
- рамки;
- меню;
- т.д.

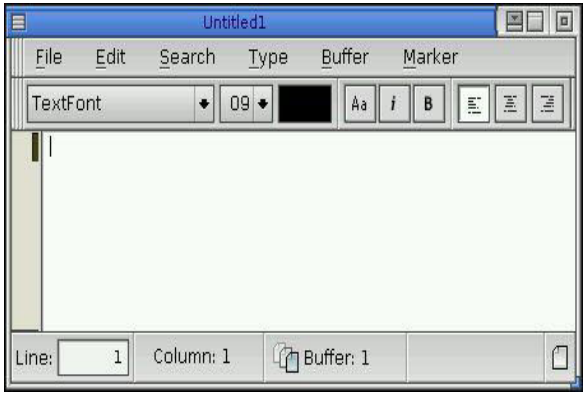


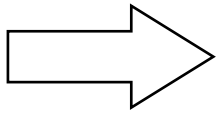
Разработка в ведется как в режиме WYSIWYG средствами штатного редактора интерфейсов оконных приложений (PhAB), так и в произвольных внешний IDE.

Поддержка библиотеки Qt в ЗОСРВ «Нейтрино»:

- 4.8.7 (штатная версия);
- 5.7.1 (экспериментальная версия, готовится к включению в состав).

Для поддержки технологий multitouch были модифицированы библиотека Qt/libqphoton.so, которая реализована в виде плагина, а также обновлен механизм интеграции Qt/Photon.





Multitouch: 2 подхода к обслуживанию

На данный момент используется два основных подхода обработки multitouch-событий:

1) Непосредственная обработка нажатий, пришедших в событии.

Плюсы:

- интуитивно понятная обработка multitouch событий;
- гибкость ввиду низкоуровневости подхода.

Минусы:

- существенное усложнение кода при увеличения сложности жеста;
- требует больше ресурсов, чем детектирование жестов.

2) Использование класса QGesture для детектирования жестов.

Плюсы:

- создание унифицированных обработчиков для жестов любой сложности;
- библиотека поддержка стандартных жестов.

Минусы:

- нестандартные жесты сложнее в реализации.

Подход 1: обработка нескольких отдельных нажатий

Основной метод классов QGraphicScene/QScene:

`bool sceneEvent(QEvent *);` – позволяет перехватывать HID события

Необходимые события:

- TouchBegin;
- TouchUpdate – позволяет отслеживать изменения координат источников;
- TouchEnd.

Пример кода, обрабатывающий multitouch-вращение:

```
const QTouchEvent::TouchPoint &tpFirst  = touchEvent->touchPoints().first();
const QTouchEvent::TouchPoint &tpSecond = touchEvent->touchPoints().last();
QLineF line1( tpFirst.lastScenePos(), tpSecond.lastScenePos() );
QLineF line2( tpFirst.scenePos(), tpSecond.scenePos() );
rotate( line2.angleTo( line1 ) );
```

Функция `rotate()` может иметь произвольную реализацию для удобства.

Подход 2: детектирование жестов (класс QGesture)

Для обработки жестов нужно вызвать метод:

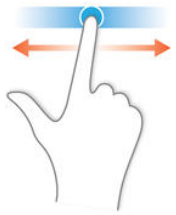
```
void grabGesture( Qt::GestureType ).
```

Пример детектирования жеста Swipe:

```
bool ImageWidget::gestureEvent( QGestureEvent *event ) {
    if ( QGesture *swipe = event->gesture( Qt::SwipeGesture ) )
        swipeTriggered( static_cast<QSwipeGesture *>( swipe ) );
    return true;
}
```

`Qt::GestureType` интерпретируется в соответствии со стандартными жестами:

Qt::PinchGesture



Qt::SwipeGesture

Qt::PanGesture

Определение собственных жестов (класс QGestureRecognizer)

Если мы хотим создать свой жест, потребуется:

- создать конфигурации жеста, который необходимо определить и обработать;
- переопределение метода `recognize()` класса `QGestureRecognizer`, который должен возвращать состояние обработки жеста (`QGestureRecognizer::Ignore`, `QGestureRecognizer::MayBeGesture`, `QGestureRecognizer::FinishGesture`, …);
- переопределение методов создания `create()` и сброса жеста `reset()`, если это необходимо;
- определение геометрии будущего жеста;
- создание экземпляра `QGestureRecognizer` и его регистрация с помощью метода `registerRecognizer()`. Удаление экземпляра при необходимости должно выполняться парным методом `unregisterRecognizer()`.

Наиболее очевидные приложения multitouch компонентов подсистемы HID:

- Картографические сервисы;
- АСУ с визуальной координацией и управлением;
- Мобильные устройства с touch-дисплеем (терминалы, планшеты, т. п.).

В ходе мастер-класса были рассмотрены следующие вопросы:

- путь HID-события от устройства и драйвера до оконного окружения и библиотеки Qt как конечного потребителя;
- актуальные методы обработки multitouch-событий в Qt;
- краткое описание различных методов детектирования и обработки жестов;
- перспективы и варианты использования multitouch.

Спасибо за внимание

Николай Коберда
Инженер-программист

+7 812 346-89-56 доб. 108
n.koberda@kpda.ru

www.kpda.ru
www.swd.ru

СВД Встраиваемые Системы

SWD Software