Лаборатория Системного ПРограммирования и ИНформационных Технологий СПбГУ

Лабораторная работа № 4
Использование веб-камеры и пальцевого интерфейса сенсорного экрана

Цель лабораторной работы

Демонстрация процесса разработки практического приложения, позволяющего рисовать пальцем на снимке с вебкамеры.

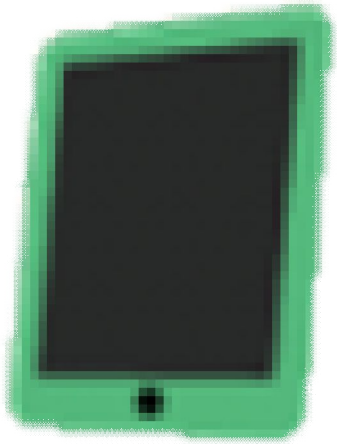
www.math.spbu.ru/user/gran/Atom21/lab04

Необходимые навыки и инструменты

- Базовые знания языка C++
- Знание основ фреймворка Qt. Желательно знакомство с библиотекой MeeGoTouch
- Представление о процессе кросс-разработки приложений
- **Устройство с тачскрином и вебкамерой — планшет под управлением** MeeGo

Постановка задачи

- Получение кадра от вебкамеры
- Отображение рисунка на элементе управления GUI
- Использование прикосновений к элементу управления для рисования на нем
- Сохранение полученного рисунка в jpeg-файл



Классы Qt для обработки изображений

- QImage — **ввод/вывод и пиксельный доступ**
- QPixmap — **отображение на экране**
- QBitmap — черно-белая (1 бит) маска
- QPicture — последовательность команд для рисования

Отображение рисунка на элементе управления GUI

```
MImageWidget widget;
QImage image;       или     QPixmap image;
widget.setImage(image);

// Присваиваем виджету размер изображения
widget.setMaximumSize(image.size());
widget.setMinimumSize(image.size());
widget.setPreferredSize(image.size());

//Получить от виджета можем только
  QPixmap!
```

Перехват событий тачскрина

Разрешить получение событий тачскина
setAcceptTouchEvents(true);
Переопределение обработчика событий

```
bool MyWidget::event(QEvent *event) {
        if (event->type() == QEvent::TouchBegin ||
            event->type() == QEvent::TouchUpdate ||
            event->type() == QEvent::TouchEnd) {
                // Фильтруем события и приводим тип
                // Не рекомендуется использовать dynamic_cast
                QTouchEvent *te =
static_cast<QTouchEvent*>(event);
                ...
                // В конце отмечаем событие как обработанное
                te->accept();
                return true;
}
```

Обработка событий тачскрина

```
if (te->type() == QEvent::TouchBegin) {
    lastPoint = te->touchPoints().first().pos();
} else {
    static QPen pen(QBrush(Qt::red),7);
    QPointF currentPoint = te->touchPoints().first().pos();
    qreal dist = distance(currentPoint, lastPoint);
    // Если точка достаточно далеко
    // или прикосновение закончилось
    if (dist > 20 || te->type() == QEvent::TouchEnd) {
        QPixmap p(*pixmap());
        // Здесь рисуем линию на QPixmap
        // Не самое эффективное решение!
        setPixmap(p);
        lastPoint = currentPoint;
    }
}
```

Отключение Gestures

Беда — тачскрин продолжает генерировать Gestures для GUI! Разрешаем получение PanGesture виджетом:

- grabGesture(Qt::PanGesture);
- Перехватываем события, ничего не делая:
- bool MyWidget::event(QEvent *event) {
- ...
- if (event->type() == QEvent::Gesture ||
- event->type() == QEvent::GestureOverride) {
- event->accept();
- return true;
- }

Сохранение измененного рисунка

Штатные средства Qt — QImageWriter и QImageReader

```
// Тип файла определен автоматически
QImageWriter jpg("/tmp/image.jpg");
// Конвертируем QPixmap в QImage и сохраняем
jpg.write(imagewidget->pixmap()->toImage());
```

Получение кадра от вебкамеры

- Используем утилиту v4l2grab из первой части курса
- Устанавливаем зависимости
- *$> zypper install libjpeg-devel*
- Собираем приложение
- *$> gcc v4l2grab.c -o v4l2grab -ljpeg*
- Размещаем в /usr/bin (нужны права root)
- *$> mv v4l2grab /usr/bin*

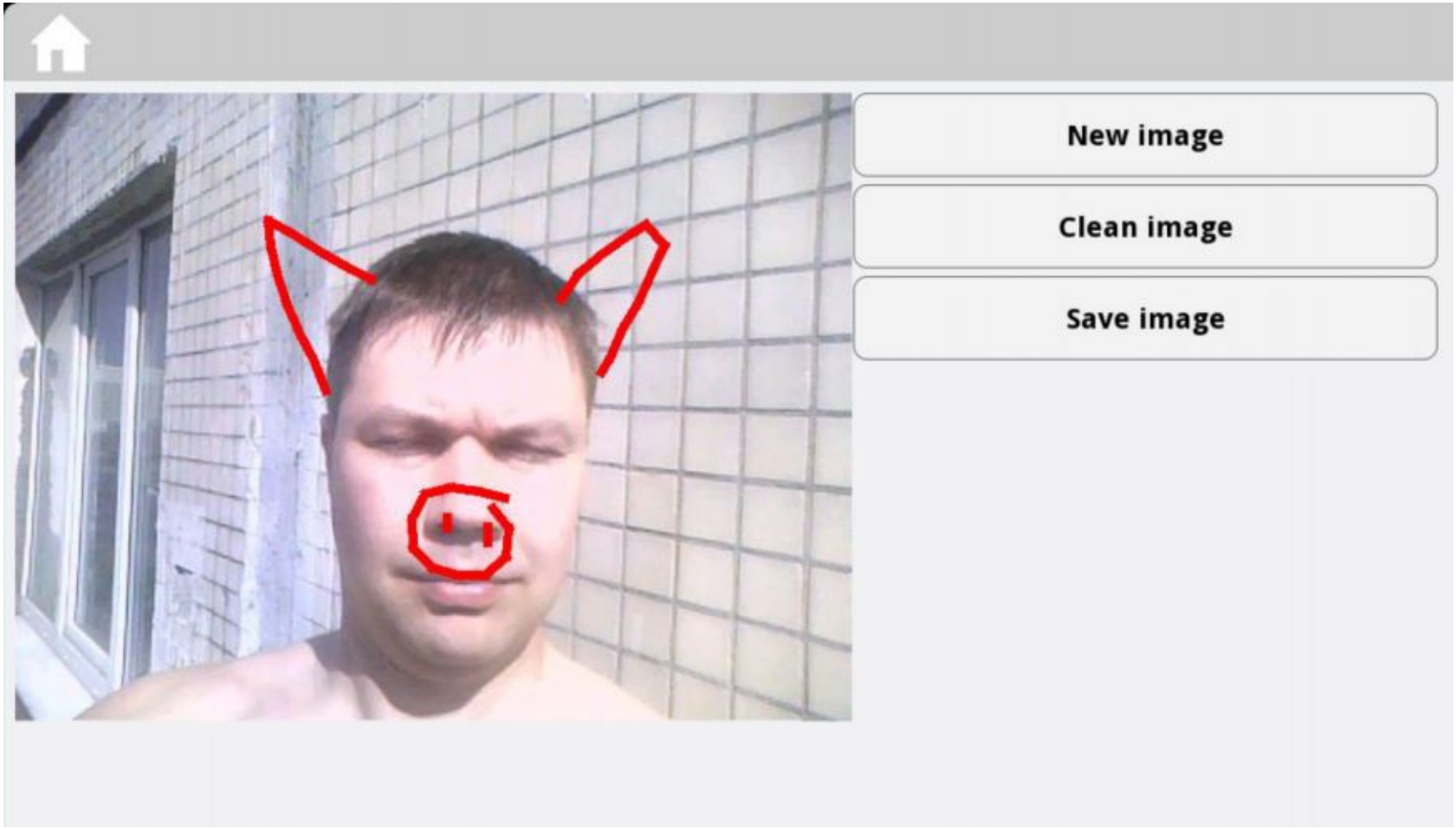
Слот для получения кадра

```
void MyWidget::createImage() {
        QProcess v4l2grab;
        // Запуск внешней программы
        v4l2grab.start("v4l2grab -W 800 -H 600 -d /dev/video0"
                        " -o /tmp/webcam_image.jpg");
        // Таймаут 3 секунды
        v4l2grab.waitForFinished(3000);
        // Читаем jpeg из файла средствами Qt
        QImageReader jpg("/tmp/webcam_image.jpg");
        jpg.read(&image);
}
```

Создание изображения программно

```
QImage image(600, 600, Qimage::Format_RGB888); // Размер изображения
QPainter* painter = new QPainter(&image);
painter->setPen(Qt::blue);                    // Параметры для вывода текста
painter->setFont(QFont("Arial", 80));
painter->drawText(image.rect(), Qt::AlignCenter,
        Qtime::currentTime().toString());     // Печатаем текущее время
painter->setPen(QPen(QBrush(Qt::red),7));     // Параметры для рамки
painter->drawLine(QPointF(0,0),QPointF(w,0)); // Рисуем рамку вокруг рисунка
painter->drawLine(QPointF(w,h),QPointF(w,0));
painter->drawLine(QPointF(w,h),QPointF(0,h));
painter->drawLine(QPointF(0,0),QPointF(0,h));
painter->end();
```

Результат работы приложения



Домашние задания

1. Добавить возможность изменять параметры пера (цвет, толщину, ...) при рисовании.
2. Добавить диалоги для открытия файла изображения с диска и записи на диск.
3. Использовать вместо утилиты v4l2grab компоненту Multimedia библиотеки Qt Mobility

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Благодарю за внимание!

Вопросы?