МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА» (НГТУ)

Институт ИРИТ

Кафедра "Вычислительных систем и технологий"

ОТЧЕТ

по прохождению производственной (технологической) практики

Направление подготовки/специальность:
09.03.01 / Информатика и вычислительная техника

Образовательная программа:
Вычислительные машины, комплексы, системы и сети

Выполнил:

Студент гр. _____
(группа) *(подпись практиканта, Ф.И.О.)*

Руководитель практики от предприятия

_____ _____
(должность) *(подпись, Ф.И.О., печать предприятия)*

Руководитель практики от кафедры
к.т.н., ст. преподаватель _____/Синенков Д.В./
(подпись)

Отчет защищен с оценкой: _____

Дата защиты «___» _____20__ г.

Н.Новгород, 20___ год

СОВМЕСТНЫЙ РАБОЧИЙ ГРАФИК (ПЛАН) ПРОВЕДЕНИЯ ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКИ

Студента гр. 19-ИВТ-3 Ф.И.О Сухорукова Валерия Алексеевича

№ п/п	Разделы (этапы) практики	Сроки выполнения с _____ по_____	Отметка о выполнении (подпись руководителя практики*)
1.	**Подготовительный (организационный) этап**	25.06.2021 - 30.06.2021	
1.1.	Проведение собрания студентов; получение индивидуального задания и путевки на практику	25.06.2021	
1.2	Оформление пропуска на предприятие	30.06.2021	
1.3	Прохождение инструктажа по технике безопасности	30.06.2021	
2.	**Производственный этап**	30.06.2021 – 18.07.2021	
2.1	Знакомство со структурой предприятия, его подразделениями, цехами, отделами	30.06.2021	
2.2	Знакомство с научно-исследовательской деятельностью предприятия.	30.06.2021	
2.3	Знакомство с организацией производственных и технологических процессов	30.06.2021	
2.4	Знакомство работой подразделения (Схемотехническое отделение, отдел 9 *)*	30.06.2021	
2.5.	Приобретение навыков работы в должности инженер-программист	30.06.2021 – 18.07.2021	
2.6.	Выполнение индивидуального задания:	1.07.2021– 18.07.2021	
	1. Ознакомление с операционной системой Astra Linux	1.07.2021	
	2. Ознакомление со средой разработки Qt - Creator	1.07.2021	
	3. Изучение литературы по разработке приложений с использованием библиотеки Qt	1.07.2021– 4.07.2021	
	4. Выполнение практического задания	5.07.2021– 18.07.2021	
3.	**Заключительный этап**	19.07.2021– 23.07.2021	
3.1	Анализ и обобщение полученной информации	19.07.2021	
3.2	Написание отчета по практике	20.07.2021– 23.07.2021	

* На этапах 1.1, 3.1, 3.2 отметку о выполнении ставит руководитель практики от кафедры, на этапах 1.2, 1.3, 2 – руководитель практики от предприятия.

Руководитель практики от кафедры

к.т.н., ст. преподаватель _____ /Синенков Д.В./

(ученые звание и степень) *(подпись)*

Руководитель практики от предприятия

_____ _____ Ф.И.О.

(должность) *(подпись)*

Введение

Цели практики

1. Ознакомление со структурой предприятия и отдела.
2. Изучение среды разработки Qt Creator.
3. Выполнение практического задания .

Цель практического задания - ознакомление со встроенными стилями приложения библиотек Qt и создание элементов собственного стиля.

Стиль приложение - это класс, реализующий возможности для рисования рамок, кнопок, растровых изображений и т.д.

Актуальность работы

Создание собственного стиля может быть обусловлено требованиями к оформлению, или необходимостью введения нового функционала.

Одним из направлений деятельности АО «ФНПЦ «ННИИРТ» является создание программного обеспечения для управления РЛС (радиолокационными станциями). Данное ПО используется в военной сфере и для него установлены ГОСТы оформления. Каждый элемент управления должен соответствовать установленным требованиям: определённая форма, цвет, размер и ряд других параметров. На текущий момент на предприятии реализован требуемый стиль приложения, но он является частью кода каждой отдельной программы. Передо мной была поставлена задача создать отдельный класс для последующего использования его в создаваемом ПО.

Задачи практики

1. Изучение структуры и реализации встроенных стилей.
2. Изучение требований к новому стилю.
3. Создание нового стиля приложения.

Основная часть отчета

Информация об организации

Полное название: Акционерное общество «Федеральный научно-производственный центр «Нижегородский научно-исследовательский институт радиотехники»

Краткое название: АО «ФНПЦ «ННИИРТ»

Организационно-правовая форма: Акционерное общество

Направления деятельности

АО «ФНПЦ «ННИИРТ» является предприятием радиотехнического профиля, выполняющим: научно-исследовательские, опытно-конструкторские разработки, техническое обслуживание, серийное изготовление в области радиолокации, радиотехнических систем и устройств.

Важными направлениями развития являются широкая модернизация радиолокационной техники, повышение эффективности технологических процессов, внедрение современной электронной компонентной базы.

Краткая история

В 1947 году на заводе № 197 имени В. И. Ленина создается специальная лаборатория по проектированию самолетной радиоаппаратуры и электрооборудования под руководством Е. В. Бухвалова.

С 1948 по 1966 разрабатывается ряд РЛС И РЛК.

В 1966 Приказом Министра радиопромышленности СССР организация переведена на самостоятельный баланс как Конструкторское бюро Горьковского телевизионного завода имени В. И. Ленина.

В 1981 Учитывая высокий научный потенциал предприятия Министерство радиопромышленности СССР преобразовало КБ ГТЗ им. В. И. Ленина в Горьковский научно-исследовательский институт радиотехники. Директором ФГУП «ГНИИРТ» назначен Владимир Алексеевич Проскурин

В 1985 впервые разработана РЛС с цифровой обработкой сигналов 1Л13. Так же в 1985 разработан РТК Э-801 для вертолета Ка-31

В 2006 Распоряжением Правительства РФ за Нижегородским научно-исследовательским институтом радиотехники сохранен статус Федерального научно-производственного центра.

В 2008 Предприятие преобразовано в Открытое акционерное общество «Федеральный научно-производственный центр «Нижегородский научно-исследовательский институт радиотехники» (ОАО «ФНПЦ «ННИИРТ»).

За всё время существования предприятие удостаивается большого числа наград и благодарственных писем.

Информация об отделе

Отдел 9 является структурным подразделением схемотехнического отделения (СТО).

Отдел занимается разработкой аппаратуры управления, синхронизации, отображения и передачи информации.

Отдел является ведущим подразделением, обеспечивающим на уровне новейших достижений науки и техники создание аппаратных и программно-аппаратных комплексов для решения следующих задач:

1. Выполнение требований тактико–технического задания на изделие по тематике отдела.
2. Разработка принципов построения систем управления, отображения и передачи радиолокационной информации.
3. Обеспечение оптимального управления режимами радиолокационных средств.
4. Эргономическое обеспечение радиолокационных средств.
5. Разработка алгоритмов и программного обеспечения для систем управления, отображения и передачи радиолокационной информации.

Охрана труда

1. Условия труда организации

Работа на предприятии проходит в требуемых условиях. Соблюдаются все нормы электро- и пожаробезопасности. В корпусах поддерживается чистота, порядок и комфортная температура воздуха.

2. Режим труда и отдыха

Режим труда и отдыха соответствует установленным правилам. При работе с ПК и ЭВМ регламентируется время нахождения перед монитором. Каждый час происходит перерыв на 10-15 минут для отдыха организма. Максимальное время, проводимое за компьютером у работника, не превышает 7 часов, у студента-практиканта – 4 часа.

3. Нормативные документы по охране труда

В Российской Федерации вопросы, относящиеся к организации и охране труда при работе за компьютером, регулируются:

- Трудовым кодексом
- «Гигиеническими требованиями к персональным электронно-вычислительным машинам и организации работы» (СанПиН 2.2.2/2.4.1340-03)
- «Типовой инструкцией по охране труда при работе на персональном компьютере» (ТОИ Р-45-084-01).

Локальные нормативные документы, действующие в организации:

- «Инструкция по охране труда для работников» (ОИТ 83)

Аппаратное и программное обеспечение, используемое в отделе

Задачей отдела является разработка программного обеспечение для систем управления, отображения и передачи радиолокационной информации. Разработка происходит на стационарных компьютерах под управлением операционных систем Windows 10 и Astra Linux. Средой разработки, используемой в отделе, является Qt-Creator.

Windows 10 — операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft.

Astra Linux — операционная система на базе ядра Linux, созданная для комплексной защиты информации и построения защищённых автоматизированных систем. Востребована в первую очередь в российских силовых ведомствах, спецслужбах и государственных органах.

Qt-Creator – кроссплатформенная среда разработки для написания приложений с графическим интерфейсом на языках C, C++.

Разработанное ПО устанавливается на устройства управления с сенсорным дисплеем ПК-31 и ПК-51 - панельный компьютер с диагональю 31 и 51 См соответственно (названия устройств внутриорганизационные).

Решение практической задачи

Внешний вид всех виджетов приложения можно изменять. Это можно делать с помощью двух механизмов: класса QStyle и таблиц стилей.

Класс QStyle - это абстрактный базовый класс, который инкапсулирует внешний вид графического интерфейса.

QStyle содержит функции для отрисовки всех элементов приложения. Основные функции:

1) void QStyle::drawItemText () - выводит текст в заданном месте.
2) void QStyle::drawPrimitive () - рисует приметив с указанными опциями стиля в указанном месте приложения.
3) void QStyle::drawControl () – рисует элемент управления приложением.
4) void QStyle::drawComplexControl () – рисует сложный элемент управления приложением.

Класс QStyle имеет подклассы QWindowsStyle, QCDEStyle, QCommonStyle и другие, в которых переопределяются функции для отрисовки всех элементов для придачи приложению определённого стиля.

Стиль приложения можно изменить, используя функцию QApplication:: setStyle().

Листинг программы для демонстрации стилей приложения

```
#include <QWidget>
#include <QPushButton>
#include <QBoxLayout>
#include <QGroupBox>
#include <QCheckBox>
#include <QRadioButton>
#include <QLabel>
#include <QLineEdit>
#include <QSlider>

#include <QApplication>

int main(int argc, char *argv[]){
    //Создание приложения
    QApplication a(argc, argv);

    //Создание основного виджета - окна приложения
    QWidget w;

    //Создание элементов управления
    QPushButton * button1=new QPushButton ("PushButton 1");
    QPushButton * button2=new QPushButton ("PushButton 2");
    QCheckBox   * button3=new QCheckBox ("CheckBox",&w);
    QRadioButton * button4 =new QRadioButton("RadioButton",&w);
    QLabel *lbl=new QLabel("common Style",&w);
    QLineEdit *line=new QLineEdit("LineEdit",&w);
    QSlider * slider=new  QSlider(Qt::Horizontal,&w);
    QHBoxLayout * phbxLayout = new QHBoxLayout;
    QGroupBox gbx("GroupBox",&w);

    //Задание связей для элемента "GroupBox"
    phbxLayout->addWidget(button1);
    phbxLayout->addWidget(button2);
    gbx.setLayout(phbxLayout);

    //Расстановка элементов в окне приложения
    gbx.move(50,120);
    button3->move(50,80);
    button4->move(50,50);
    lbl->move(110,10);
    line->move(50,190);
    slider->move(50,30);

    //Задание размера окна
    w.setMinimumSize(300,230);

    //Установка стиля приложения
    a.setStyle("common");

    //Отображение окна
    w.show();

    //Завершение работы
    return a.exec();
}
```

При запуске программы отображаются все элементы в стиле, заданном параметром в функции setStyle. Параметром является строковая константа : common, windows, cde и другие.

При передаче параметра common приложение примет следующих вид:

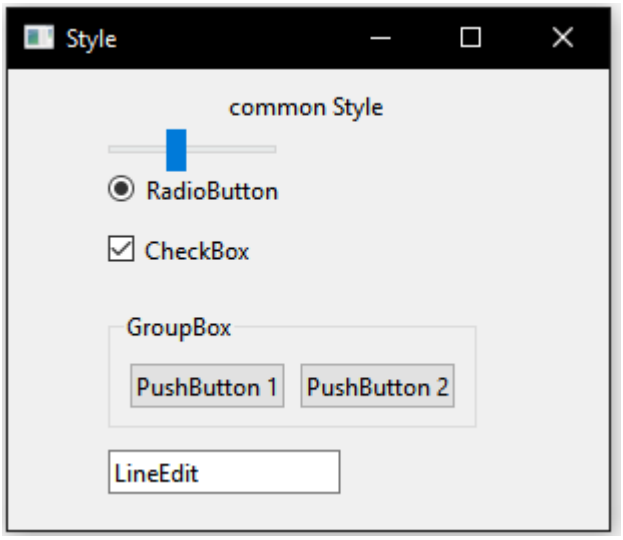


Рисунок 1. Стиль приложения common

Изменяя параметр на windows и fusion получим следующие результаты:

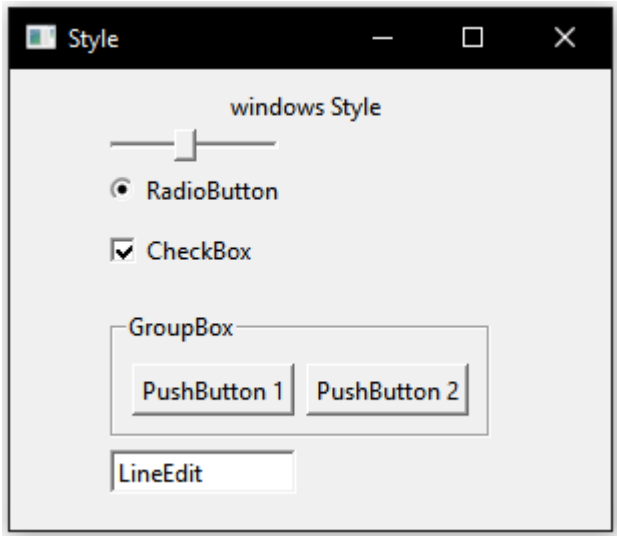


Рисунок 2. Стиль приложения windows

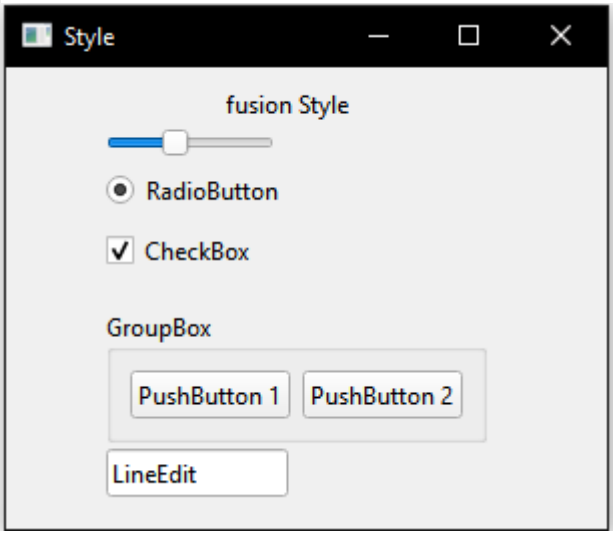


Рисунок 3. Стиль приложения fusion

Таблицы стилей Qt - мощный механизм, который позволяет настраивать внешний вид виджетов, в дополнение к тому, что уже доступен с помощью создания подклассов QStyle. Концепция, терминология и синтаксис таблиц стилей Qt в значительной части вдохновлены Каскадными таблицами стилей (Cascading Style Sheets, CSS) HTML, но адаптированы к миру виджетов.

Таблицы стилей - текстовые спецификации, которые могут быть установлены для всего приложения с помощью QApplication::setStyleSheet() или для определенного виджета (и его потомков) посредством QWidget::setStyleSheet(). Если на различных уровнях установлено несколько таблиц стилей, то Qt порождает эффективную таблицу стилей изо всех установленных таблиц стилей. Это называется каскадированием.

Например, следующая таблица стилей определяет, что все объекты QLineEdit должны использовать в качестве фона желтый цвет, а все объекты QCheckBox должны использовать красный как цвет текста:

```
QLineEdit {background: yellow}
QCheckBox {color: red}
```

Цель практического задания – ознакомление с классом QStyle и его подклассами и изучение их строения. С помощью полученных знаний создать подкласс класса QCommonStyle для реализации стиля NNIIRT_Style.

В ходе выполнения практического задания были переопределены функции void QStyle::drawControl () и void QStyle::drawComplexControl () для отрисовки следующих элементов: простоя кнопка, кнопка с фиксацией нажатия, кнопка-табло, группа кнопок.

Для переопределения функций был создан класс NNIIRT_Style унаследованный от QCommonStyle. Его объявление находится в файле nniirt_style.h (приложение 1), реализация в файле nniirt_style.cpp (приложение 3).

```
class NNIIRT_Style : public QCommonStyle{}
```

Для использования стиля необходимо использовать функцию setStyle с указателем на класс NNIIRT_Style.

```
a.setStyle(new NNIIRT_Style());
```

Простоя кнопка – элемент управления, присутствующий в библиотеках Qt, при нажатии на который в программу поступает одиночный сигнал, на который она как-либо реагирует.

Кнопка с фиксацией нажатия - элемент управления, присутствующий в библиотеках Qt, при нажатии на который в программу поступает непрерывный сигнал, который прекращается при повторном нажатии на кнопку. Элемент используется для выполнения какого-либо действия при присутствии сигнала.

Кнопка-табло – созданный элемент управления. Используется для открытия вложенного окна управления приложением. При нажатии на кнопку открывается дополнительное окно.

Элемент создан для реализации нового функционала. Среди существующих элементов нет такого, при нажатии на который открывается новое окно.

Для реализации был создан класс button_with_window наследованный от QCheckBox. Его объявление находится в файле button_with_window.h (приложение 2), реализация в файле button_with_window.cpp (приложение 4).

```
class Button_with_window : public QCheckBox{}
```

Создание экземпляра класса:

```
Button_with_window * button3 = new Button_with_window ("Название",&second_w,&w);
```

Группа кнопок - элемент управления, присутствующий в библиотеках Qt. Служит для объединения кнопок по какому-либо признаку.

Демонстрация элементов с применённым созданным стилем

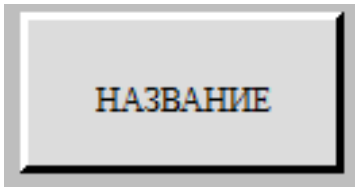


Рисунок 4. Простая кнопка

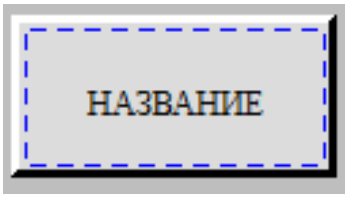


Рисунок 5. Кнопка с фиксацией нажатия

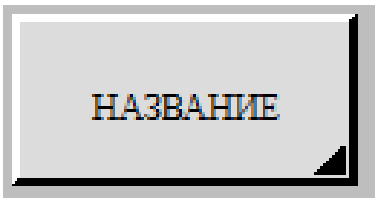


Рисунок 6. Кнопка-табло

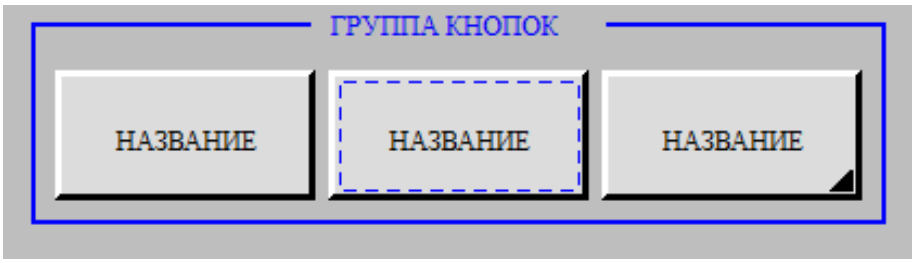


Рисунок 7. Группа кнопок

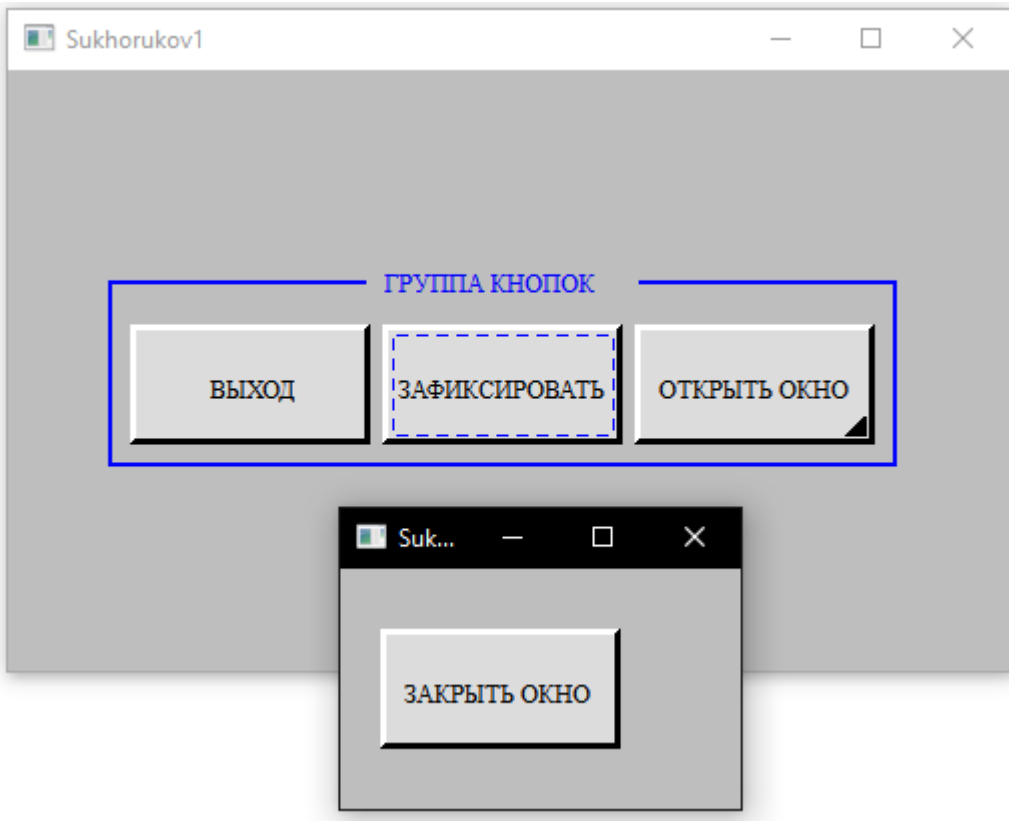
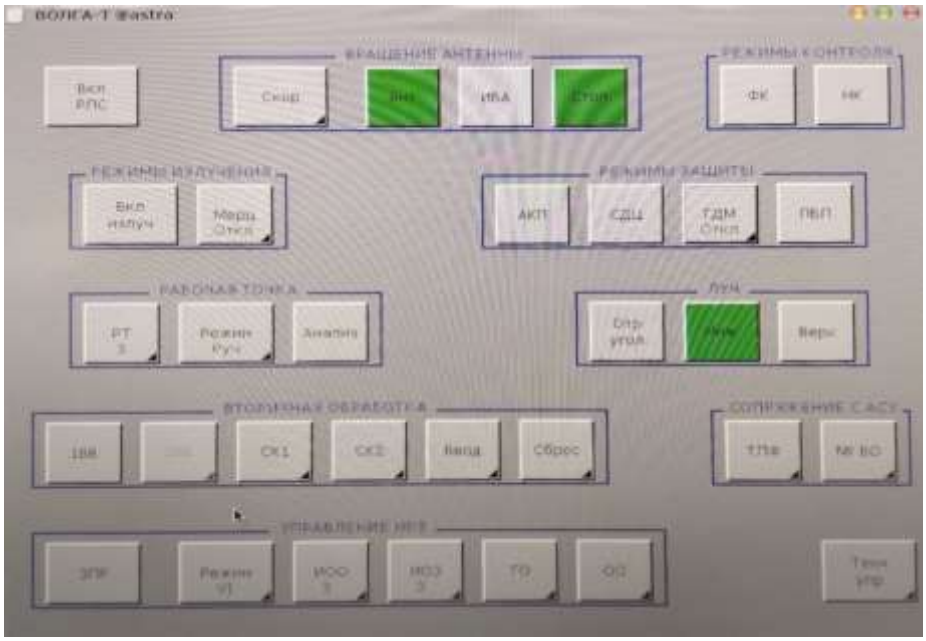


Рисунок 8. Демонстрация работы приложения

Так же во время прохождения практики было установлено тестовое программное обеспечение для управления радиолокационными станциями на базе операционной системы Astra Linux на устройство управления ПК-31 (сенсорный панельный компьютер с диагональю 31 см.).



Заключение

Выполнение практического задания стало интересным и познавательным применением, полученных в ходе обучения, навыков и знаний. Прикладное применение навыков программирования – полезный опыт.

В ходе практики было изучено и освоено:

1. Тематика работы предприятия и отдела 9
2. Среда разработки Qt-Creator
3. Выполнено практическое задание.

Список использованных источников

1. Qt 4.8. Профессиональное программирование на C++. — СПб. БХВ-Петербург, 2012
2. https://doc.qt.io/archives/qtjambi-4.5.2_01/

Приложения

Приложение 1.

Код файла nniirt_style.h

```
#ifndef NNIIRT_STYLE_H
#define NNIIRT_STYLE_H

#include <QtGui>
#include <QCommonStyle>
#include <QStyleOption>
#include<QPushButton>
#include <QBoxLayout>
#include <QGroupBox>

/* Класс стиля приложения.
   Родительский класс- Стандартный стиль.
*/
class NNIIRT_Style : public QCommonStyle{
    Q_OBJECT
public:
    //Конструктор без параметров
    NNIIRT_Style();

    //Функция отрисовки простого элемента управления приложением
    void drawControl(QStyle::ControlElement element, const
                     QStyleOption *opt, QPainter *p, const QWidget
                     *w) const;
    //Функция отрисовки сложного элемента управления приложением
    void drawComplexControl(QStyle::ComplexControl cc, const
                            QStyleOptionComplex *opt, QPainter *p,
                            const QWidget *w ) const;
};
#endif // NNIIRT_STYLE_H
```

Приложение 2.

Код файла button_with_window.h

```
#ifndef BUTTON_TO_SHOW_WINDOW_H
#define BUTTON_TO_SHOW_WINDOW_H

#include <QCheckBox>
#include <QWidget>

/* Класс кнопки, при нажатии на которую открывается новое окно.
 * Родительский класс - кнопка-флаг (кнопка с проверкой нажатия)
*/
class Button_with_window : public QCheckBox{
public:
    //Конструктор без параметров
    Button_with_window();
    //Конструктор с параметрами
    Button_with_window (QString, QWidget*, QWidget*);

    //Функция для отображения окна
    void show_window();

    //Указатель на виджет - окно
    QWidget* window;

};

#endif // BUTTON_TO_SHOW_WINDOW_H
```

Приложение 3.

Код файла nniirt_style.cpp

```
#include "nniirt_style.h"
#include "button_with_window.h"

/*-------------------------Constructor-------------------------*/
NNIIRT_Style::NNIIRT_Style() :
    QCommonStyle(){}



/*-------------------------drawControl-----------------------*/
/* Функция отрисовки простого элемента управления приложением
 * Параметры функции:
 *  1) Константа для определения типа элемента.
 *  2) Указатель на стилевые параметры элемента.
 *  3) Указатель на "художника", ответственного за отрисовку
элементов.
 *  4) Указатель на виджет - элемент, который необходимо
отрисовать.
 * Результат работы функции - элемент, нарисованный в окне
приложения.
*/
void NNIIRT_Style::drawControl(QStyle::ControlElement element,
const QStyleOption *opt, QPainter *p, const QWidget *w) const{
    switch( element ){
        //Секция отрисовки "нажимной" кнопки
        case CE_PushButton:{
             QPushButton * button=(QPushButton*)w;

             //Отрисовка кнопки прямоугольной формы
             p->fillRect(0,0,button->width(),
                   button->height(),QColor( 220, 220, 220 ));

             //Отрисовка левой и верхней частей рамки кнопки
             p->setPen( QPen( QColor( 255, 255, 255)));
             for(int i=0;i<3;i++){
                p->drawLine(QPoint(i,0),QPoint(i,
                                        button->height()-1));
                p->drawLine(QPoint(0,i),QPoint(
                                        button->width()-1,i));
             }

             //Отрисовка правой и нижней частей рамки кнопки
             p->setPen( QPen( QColor( 0, 0, 0)));
             for(int i=0;i<3;i++){
                 p->drawLine(QPoint(0+i,button->height()-i-1),
                             QPoint(button->width()-1,
                                      button->height()-i-1));


                 p->drawLine(QPoint(button->width()-i-1,i),
```

```
                        QPoint(button->width()-i-1,
                               button->height()-1));
            }

            //Вывод текста кнопки
            QStyleOptionButton* opt_but=(QStyleOptionButton*)opt;
            int x=(opt_but->rect.width()-
                            (opt_but->text.size())*8)/2;
            int y=(opt_but->rect.height()-8)/2;
            QRect textRect (QPoint(x,y),QPoint(
                            opt_but->rect.width(),
                            opt_but->rect.height()));
            p->setPen( QPen( QColor( 0, 0, 0 )));
            p->setFont(QFont("times new roman"));
            p->drawText(textRect,opt_but->text);

            //Если кнопка обладает свойством фиксации состояния
            //нажатия, то в случае нажатия отрисовывается
            //пунктирная рамка
            if(button->isCheckable()){
                if(button->isChecked()){
                    p->setPen( QPen( QColor( 0, 0, 255)));
                    //Отрисовка правой и левой частей рамки
                    for(int i=7;i<button->height()-5;i=i+10){
                        p->drawLine(QPoint(5,i),QPoint(5,i+5));
                        p->drawLine(QPoint(button->width()-5,i),
                                    QPoint(button->width()-5,i+5));
                     }
                    //Отрисовка верхней и нижней частей рамки
                    for(int i=7;i<button->width()-5;i=i+10){
                        p->drawLine(QPoint(i,5),QPoint(i+5,5));
                        p->drawLine(QPoint(i,button->height()-5),
                                    QPoint(i+5,button->height()-5));
                     }
                }
            }
        }
        break;
        // Конец секции отрисовки "нажимной" кнопки

        //Секция отрисовки кнопки, при нажатии на которую
открывается новое окно
        case CE_CheckBox:{
                    Button_with_window* check_but
                                    =(Button_with_window*)w;

               //Отрисовка кнопки прямоугольной формы
               p->fillRect(0,0,check_but->width(),
                  check_but->height(),QColor( 220, 220, 220 ));

               //Отрисовка левой и верхней частей рамки кнопки
               p->setPen( QPen( QColor( 255, 255, 255)));
               for(int i=0;i<3;i++){
```

```
    p->drawLine(QPoint(i,0),
                QPoint(i,check_but->height()-1));
    p->drawLine(QPoint(0,i),
                QPoint(check_but->width()-1,i));
}

//Отрисовка правой и нижней частей рамки кнопки
p->setPen( QPen( QColor( 0, 0, 0)));
for(int i=0;i<3;i++){
    p->drawLine(QPoint(0+i,check_but->height()
                -i-1),
                QPoint(check_but->width()
                -1,check_but->height()-i-1));

    p->drawLine(QPoint(check_but->width()
                -i-1,i),
                QPoint(check_but->width()
                -i-1,check_but->height()-1));
}

//Отрисовка треугольника в правом нижнем углу
for(int i=0;i<10;i++){
    p->drawLine(check_but->width()-5,
                check_but->height()-5-i,
                check_but->width()-15+i,
                check_but->height()-5-i);

}

//Вывод текста кноки
QStyleOptionButton* opt_but
                      =(QStyleOptionButton*)opt;

int x=(opt_but->rect.width()
                -(opt_but->text.size())*8)/2;
int y=(opt_but->rect.height()-8)/2;

QRect textRect (QPoint(x,y),
                QPoint(opt_but->rect.width(),
                       opt_but->rect.height()));
p->setPen( QPen( QColor( 0, 0, 0 )));
p->setFont(QFont("times new roman"));
p->drawText(textRect,opt_but->text);

//Если кнопка нажата, то отображается новое окно
if(check_but->isChecked()){
   //Отображение дочернего окна
   check_but->show_window();
   //Перевод кнопки в состояние "отжата"
   check_but->setChecked(false);
   //Изменение цвета заднего фона нового окна
   QColor background( 190, 190, 190 );
   check_but->window->setAutoFillBackground(true);
```

```
                    check_but->window->setPalette(background);
                }

        }
        break;
        // Конец секции отрисовки кнопки, при нажатии на которую
        // открывается новое окно

        //Для отображения не переопределённых элементов,
         //вызывается функции
        //родительского класса - стандартного стиля
        default:
            QCommonStyle::drawControl(element, opt, p, w);
        break;
    }
}


/*---------------------drawComplexControl--------------------*/
/* Функция отрисовки сложного элемента управления приложением
 * Параметры функции:
 *  1) Константа для определения типа элемента.
 *  2) Указатель на стилевые параметры элемента.
 *  3) Указатель на "художника", ответственного за отрисовку
элементов.
 *  4) Указатель на виджет - элемент, который необходимо
отрисовать.
 * Результат работы функции - элемент, нарисованный в окне
приложения.
*/
void NNIIRT_Style::drawComplexControl(QStyle::ComplexControl cc,
const QStyleOptionComplex *opt, QPainter *p, const QWidget *w )
const{
    switch(cc){
        //Секция отображения группы кнопок
        case CC_GroupBox:{
            QGroupBox * group=(QGroupBox*)w;
            //Изменение цвета пера
            p->setPen( QPen( QColor( 0, 0, 255 )));

            //Отрисовка левой границы, толщиной 2 пикселя
            p->drawLine(0,5,0,group->height()-1);
            p->drawLine(1,5,1,group->height()-1);

            //Отрисовка нижней границы, толщиной 2 пикселя
            p->drawLine(0,group->height()-1,group->width()
                        -1,group->height()-1);
            p->drawLine(0,group->height()-2,group->width()
                        -1,group->height()-2);



            //Отрисовка правой границы, толщиной 2 пикселя
```

```
            p->drawLine(group->width()-1,5,group->width()-1,
                        group->height()-1);

            p->drawLine(group->width()-2,5,group->width()-2,
                         group->height()-1);

            QStyleOptionGroupBox *opt_group
                                    =(QStyleOptionGroupBox*)opt;

            //Определение ширины текста
            int text_width=opt_group->text.size()*9;

            //Отрисовка левой части верхней границы
            p->drawLine(0,5,(group->width()-text_width)/2-10,5);
            p->drawLine(0,6,(group->width()-text_width)/2-10,6);

            //Отрисовка правой части верхней границы
            p->drawLine(group->width()-1,5,
                         (group->width()+text_width)/2+10,5);
            p->drawLine(group->width()-1,6,
                         (group->width()+text_width)/2+10,6);

            //Вывод названия группы кнопок
            QRect textRect ((group->width()-text_width)/2,
                            0,text_width,20);

            p->setPen( QPen( QColor( 0, 0, 255 )));
            p->setFont(QFont("times new roman"));
            p->drawText(textRect,opt_group->text);
        }
        break;
        //Конец секции отображения группы кнопок

        //Для отображения не переопределённых элементов,
         //вызывается функция родительского класса
         //- стандартного стиля
        default:
            QCommonStyle::drawComplexControl(cc,opt,p,w );
        break;
    }
}
```

Приложение 4.

Код файла button_with_window.cpp

```
#include "button_with_window.h"

/*--------------------Constructor----------*/
Button_with_window::Button_with_window():
    QCheckBox (){}

/*--------------------Constructor----------*/
Button_with_window::Button_with_window (QString s, QWidget *w,
QWidget*p) {
   this->window=w;
   this->setText(s);
   this->setParent(p);
   this->setMouseTracking(true);
}


/*-------------------------show_window-------------------------*/
/* Функция отображения окна
 * Результат работы функции - отображение дочернего окна
*/
void Button_with_window::show_window(){
        this->window->show();
}
```