11. Лабораторная работа № 5 «Обеспечение положительного User Expirience/Usability в сложных пользовательских интерфейсах»

11.1. Цель лабораторной работы

Демонстрация процесса разработки практического приложения. Отображение элементов управления пользовательского интерфейса программы.

11.2. Введение

User Experience (опыт пользователя, UX) – неотъемлемая составляющая разработки программного продукта для массовой аудитории. Не обеспечив положительный опыт пользователя, нельзя добиться успеха на рынке.

Согласно стандарту ISO 9241-210 User Experience – «ощущения и реакция человека, вследствие использования или предполагаемого использования продукта, системы или услуги». UX характеризует не саму программу или ее пользовательский интерфейс, а восприятие его пользователем, т.е. необходимо также учитывать предыдущий опыт пользователя и контекст использования. Причем имеет значение не только факт использования программы, но и отказ от использования по каким-либо причинам тоже.

Обеспечение положительного опыта использования – задача не столько программиста, сколько специалистов по маркетингу, usability и т. п., они выступают постановщиками задачи для программиста. Цель программиста – оптимально использовать возможности платформы для выполнения этой задачи, в первую очередь путем создания адекватного пользовательского интерфейса.

Для создания приложений в Linux достаточного использования библиотеки *Qt*. Приложение будет вполне работоспособным, но его пользовательский интерфейс придется вручную адаптировать к особенностям Lixux. Чтобы избежать такой адаптации, создана специализированная библиотека *Qt* – MeeGoTouch, – пользовательские интерфейсы в которой уже адаптированы.

11.3. Инструкция по выполнению лабораторной работы

Наша задача – научиться работать со специфическими элементами пользовательского интерфейса библиотеки MeeGoTouch. Для этого необходимы базовые знания языка C++ и знакомство с основами фреймворка *Qt*. В качестве целевого устройства мы использовали MeeGo 1.1.99 Tablet на планшете 3Q TU1102T, но также возможно использовать виртуальную машину или эмулятор.

Мы рассмотрим следующие элементы управления: страницы и списки. Для самостоятельно изучения можно порекомендовать разные виды кнопок (`MButton`), систему меню (`MApplicationMenu`), а также работу с панелью инструементов (`tool bar`).

Страница – контейнер для других элементов управления, аналог закладок в традиционных пользовательских интерфейсах. Ее появление вызвано тем, что на мобильных устройствах экраны небольшого размера, а элементы управления, наоборот, крупные, рассчитанные на работу с пальцами. На одном экране невозможно разместить большое количество элементов, поэтому приходится разносить их на разные страницы. Альтернативный подход – экран с прокруткой, который можно «сдвигать» в поисках нужного элементы, этот подход мы рассматривать не будем.

В MeeGoTouch страницы реализованы классом `MApplicationPage`, есть встроенная функциональность по управлению их отображением. Для того, чтобы страница

отобразилась на экране, достаточно вызвать для нее метод `appear()`, например, в обработчике нажатия на кнопку:

```
void MyApp::switchPage2() {
    page2.appear(&window);
}
…
connect(but,SIGNAL(clicked()),this,
SLOT(switchPage2()));
```

При этом одномоментно на экране может быть только одна страница. Программист сам решает, в какой момент это должно быть сделано, а система ведет учет открытых страниц и предоставляет пользователю кнопку «Back» для возврата.

Для навигации между страницами могут быть использованы различные схемы. Простейший пример приведен в этой лабораторной работе, когда переключение между страницами осуществляется нажатием кнопки на самой странице. Такой подход годится для каких-то вложенных настроек или в случае последовательной работы какого-либо мастера. Другой способ – разместить кнопки управления в области «Tap Bar», где они будут всегда доступны пользователю и он сможет вызвать желаемую страницу в любой момент. Минус в том, что навигационные кнопки постоянно занимают дефицитное место на экране. Еще один способ – использовать меню приложения для переключения страниц.

Один из наиболее сложных элементов управления – список. Во-первых, он предназначен для работы с большим количеством данных, во-вторых, списки в *Qt* традиционно сделаны довольно громоздко. В MeeGoTouch список реализован классом MList, в нем по умолчанию задействована только одна колонка, это вполне уместно для небольшого экрана мобильного устройства.

Для того, чтобы увидеть свой список на экране, программист должен реализовать как минимум два класса – класс с описанием данных для списка и класс, отвечающий за визуализацию ячеек. Описание этих классов желательно размещать в отдельном файле. Такая сложная процедура унаследована из *Qt* и кажется слишком тяжеловесной – в Android для создания списка достаточно указать ему массив строковых значений.

Класс с описанием данных наследуется от `QAbstractListModel`, в нем достаточно переопределить функцию

rowCount(), которая возвращает количество строк в списке, и собственно функцию для данных:

```
QVariant ListModel::data(const QModelIndex &index,
int role) const {
  if (role == Qt::DisplayRole) {
      QStringList rowData;
      rowData << QString("Row %1").arg(index.row());
      rowData << "some details";
      return QVariant(rowData);
  }
  return QVariant();
}
```

Отметим, что для передачи данных используется QVariant – тип-контейнер, в который можно «положить» значения разных типов. В данном случае используется QStringList, но можно использовать и свою структуру данных – главное, чтобы это было согласовано с классом-визуализатором.

Визуализатор ячеек унаследован от класса MAbstractCellCreator<MContentItem>. Обратите внимание на нетипичное для Qt использование шаблона – этот класс не является наследником QObject! MContentItem – специальный виджет, предназначенный для отображения отдельной ячейки списка, именно он занимается отрисовкой упомянутых выше двух строк и картинки. При более глубокой работе со списком программист может заменить этот класс на свой собственный, более продвинутый. В простейшем же случае достаточно переопределить одну фунцию в MAbstractCellCreator<MContentItem>:

```
void CellCreator::updateCell(const QModelIndex&
index, MWidget *cell) const
{
    MContentItem *contentItem =
qobject_cast<MContentItem *>(cell);
    QVariant data = index.data(Qt::DisplayRole);
    QStringList rowData = data.value<QStringList>();
    contentItem->setTitle(rowData[0]);
    contentItem->setSubtitle(rowData[1]);
    contentItem-
>setImage(makeImage(50,50,rowData[0],8,Qt::red));
}
```

Как можно видеть, этот метод работает с тем самым `QVariant`, который генерирует предыдущая функция. Более правильным было бы упаковать в него и `QImage`, но мы этого не сделали для наглядности материала.

11.4. Задания для самостоятельной работы

1. Создайте такие элементы пользовательского интерфейса как: разные виды кнопок (MButton), систему меню (MApplicationMenu), а также tool bar (tool bar).
2. Создать *Qt*-приложение без использования MeeGoTouch, так чтобы его пользовательский интерфейс реагировал на датчик ориентации.

Список литературы

1. Handset UI Guidelines. https://meego.com/developers/ui-design-guidelines/handset