<https://www.youtube.com/watch?v=rGxDZPYcWIs&list=PLfIJKC1ud8giOsk-C4BCOwSHtbXqTNb1W>

# TouchGFX

Добавляем элемент взаимодействия (interactions) элемента на экране.

Определяем название виртуальной функции для реализации взаимодействия.

Генерируем код.

# IDE

Проект/Application/User/generated/Screen1ViewBase.cpp – этот файл генерирует TouchGFX, его изменять нельзя. В этом файле появляются описания кнопок и ссылки на их функции взаимодействия (**void** **Screen1ViewBase::buttonCallbackHandler)**.

Код функции взаимодействия нужно прописать в файле Проект/Application/User/gui/Screen1View.cpp. Пользовательские изменения в этом файле сохраняются.

Для начала нужно в Screen1View.cpp открыть заголовочный файл Screen1View.hpp, а в нём декларировать функцию в классе **class** Screen1View : **public** Screen1ViewBase. Сохранить изменения.

Теперь можно вернуться в Screen1View.cpp и прописать функцию.

Возможно, понадобится **#include** "stm32f4xx\_hal.h"

Компилировать и запустить.

# Создаём колесо прокрутки в TouchGFX

Колесо прокрутки реализовано в TouchGFX в виде контейнера.

Контейнер – это объект, в который могут вложены текстовые поля, графические элементы, фоновые изображения, другие контейнеры и т.п.

Сначала создаются пользовательские контейнеры-шаблоны, которые могут быть использованы при создании колёс прокрутки на разных экранах.

Затем на конкретном экране создаётся специальный объект TouchGFX – колесо прокрутки, который тоже является контейнером. В созданном контейнере-колесе прокрутки можно использовать пользовательские контейнеры-шаблоны.

Рисунок . Формирование ScrollWheel контейнера в TouchGFX.

Select item

Текст

Изображение

Текст

Изображение

Пользовательский контейнер 1

Пользовательский контейнер 2

Wheel items

ScrollWheel TouchGFX контейнер

Контейнеры-шаблоны могут быть использованы как шаблоны для наполнения колеса прокрутки элементами, так и в качестве шаблонов для наполнения выбранного элемента в колесе прокрутки. Обычно контейнер колеса прокрутки включает оба шаблона: и шаблон для наполнения колеса прокрутки, и шаблон выбранного элемента в колесе.

Если в колесе прокрутки должен присутствовать элемент-переменная в текстовом формате, то в контейнер-шаблон помещается тестовое поле с wildcard, в котором определяются идентификатор текстовой переменной, размер текстового буфера.

# Генерация кода пользовательских контейнеров

TouchGFX генерирует два типа кода в разных папках:

* + 1. код, который нельзя модифицировать, поскольку при очередной генерации кода он затрёт все изменения пользователя в папке TouchGFX/generated/,
    2. код для внесения пользовательского кода в папке TouchGFX/gui/.
    3. Далее рассматриваем папки, вложенные в указанные две папки.

TouchGFX генерирует код, в котором базовые описания пользовательских контейнеров записаны в папке gui\_generated/include/gui\_generated/containers/. Имя файла описания контейнера формируется как <Имя\_контейнера>Base.hpp.

Код описания контейнеров, предназначенный для изменений пользователем, будет в папке include/gui/containers/. Имя файла описания контейнера формируется как <Имя\_контейнера>.hpp В описании контейнера обязательно есть ссылка на базовый контейнер <Имя\_контейнера>Base.hpp.

В базовом описании контейнера TouchGFX автоматически создаст класс <Имя\_контейнера>Base. В этот класс TouchGFX включит заданные пользователем параметры контейнера.

Собственно программные коды располагаются в папках

* + 1. gui\_generated/src для базового кода,
    2. src для пользовательского кода.

Рисунок . Генерация кодов для колеса прокрутки.

ScreenBase

Screen

Container  
Base

Container

Container  
Selected  
Base

Container  
Selected

TouchGFX/generated/ - TouchGFX generated

TouchGFX/gui/ - User generated

Пользователь должен в своей пользовательской части заголовка и в коде описания контейнера создать функцию для обновления элемента списка контейнера.

В дальнейшем эта функция будет использована в работе колеса прокрутки с пользовательским контейнером на экране.

# Генерация кода колеса прокрутки на экране

TouchGFX генерирует коды базовый и пользовательский для экрана аналогично генерации кодов для контейнеров.

В базовом описании экрана TouchGFX сгенерирует коды просмотровщика (View) экрана, в котором будут объявлены параметры колеса прокрутки и две функции:

* + 1. …UpdateItem() – функция заполнения видимой части колеса прокрутки значениями,
    2. …UpdateCenterItem() – функция заполнения значением выбранной (selected) части колеса прокрутки.

Функция UpdateItem всегда генерируется для колеса прокрутки и может быть реализована для обновления содержащихся элементов, в то время как функция UpdateCenterItem обновляет элемент на основе шаблона стиля выбранного элемента и, следовательно, генерируется только в том случае, если выбрано использование шаблона стиля выбранного элемента. Помимо обновления различных элементов, две функции содержат те же параметры, которые используются для обновления элементов в колесе прокрутки.

Параметр itemIndex содержит значение индекса элемента, которое используется для определения того, какой элемент колеса прокрутки обновляется. Параметр item является ссылкой на видимый элемент (контейнер) в колесе прокрутки.

При описании функций Update понадобится функция для преобразования itemIndex в отображаемый в колесе прокрутки текст. Такую функцию пользователь должен создать самостоятельно в пользовательской части пользовательского контейнера. Объявление этой функции разместить в классе с именем контейнера в заголовке hpp, а описание – в теле кода cpp.

То же самое нужно сделать и для отображения текста в выбранном элементе колеса прокрутки. В данном случае функция объявляется и описывается в коде пользовательского контейнера с описанием выбранного элемента.

Для отображения текстового значения используются переменные тестовых полей, внедрённых в контейнеры.

# Создание пользовательского кода колеса прокрутки

Перейти в базовый заголовочный файл экрана с описанием базового класса прямо из заголовочного файла просмотровщика (View) экрана по ссылке #include.

Скопировать объявления функций ScrollWheel…UpdateItem и ScrollWheel…UpdateCenterItem из базового класса, например, Settings6ViewBase, в пользовательский класс Settings6View соответственно.

Создать описание скопированных функций класса в коде страницы в файле cpp.

Создать callback для обработки события в колесе.

Создать функцию обработки события в колесе (…Handler). Объявить эту функцию в классе.

Создать в TouchGFX interactions функцию открытия колеса при нажатии на кнопку. TouchGFX сгенерирует объявление этой функции в базовый заголовочный файл экрана.

Скопировать объявление функции в пользовательский заголовочный файл.

Создать функцию открытия колеса при нажатии на кнопку в пользовательском коде.

# Callback

touchgfx/Callback.hpp, https://support.touchgfx.com/docs/api/classes/structtouchgfx\_1\_1\_callback

Обратный вызов — это, по сути, оболочка функции-указателя на член. Он используется для регистрации обратных вызовов между виджетами. Например, [кнопка](https://support.touchgfx.com/docs/api/classes/classtouchgfx_1_1_button) может быть настроена на вызов функции-члена при нажатии на нее.

Класс создан по шаблону, чтобы указать тип класса объекта, в котором находится функция-член, и типы аргументов вызываемой функции.

Класс [обратного вызова](https://support.touchgfx.com/docs/api/classes/structtouchgfx_1_1_callback) существует в четырех версиях для поддержки функций-членов с 0, 1, 2 или 3 аргументами. Компилятор автоматически определит, какой тип использовать.

Параметры шаблона:

* + 1. dest\_type - тип класса, в котором находится функция-член.
    2. T1 - тип первого аргумента в функции-члене или void, если его нет.
    3. T2 - тип второго аргумента в функции-члене или void, если его нет.
    4. T3 - тип третьего аргумента в функции-члене или void, если его нет.

**Примечание**: Вызываемая функция-член должна возвращать void. Функция может иметь ноль, один, два или три аргумента любого типа.

**Inherits from**: [GenericCallback< void, void, void >](https://support.touchgfx.com/docs/api/classes/classtouchgfx_1_1_generic_callback)

@file touchgfx/Callback.hpp declares the touchgfx::GenericCallback and touchgfx::Callback classes.

D:\ST\Defrost\Middlewares\ST\touchgfx\framework\include\touchgfx

Generic Callback - это базовый класс для обратных вызовов.

Причина, по которой существует этот базовый класс, заключается в том, что для обычного обратного вызова требуется, чтобы был известен тип класса, в котором находится функция обратного вызова. Это проблематично для т.н. фреймворковые виджеты, такие как AbstractButton, в которых должна быть возможность зарегистрировать обратный вызов для типов объектов, которые зависят от пользователя и, следовательно, неизвестны AbstractButton.

\* \* \* \* Это решается тем, что вместо этого AbstractButton содержит указатель на общий обратный вызов. Затем этот указатель должен быть инициализирован, чтобы указывать на экземпляр обратного вызова, созданный пользователем, который инициализируется соответствующим типом объекта.

\*

\* @\* @param T1 Тип первого аргумента в функции-члене или void, если нет.

\* @\* @param T2 Тип второго аргумента в функции-члене или void, если нет.

\* @\* @param T3 Тип третьего аргумента в функции-члене или void, если нет

Функции обработки событий в анимации колеса прокрутки описаны в классе Callback. Функции написаны для вызова из широкого перечня объектов, тип которых заранее неизвестен. Чтобы правильно отработать функцию обработки события в анимации колеса прокрутки, используется смарт-пойнтер на функцию обработки, в который с помощью шаблона можно передать ссылку на вызывающий объект.

Порядок создания обратного вызова:

* + 1. В классе текущего окна в разделе private объявляется смарт-пойнтер класса callback с нужным шаблоном.
    2. В классе текущего окна в разделе private объявляется прототип функции обработки обратного вызова.
    3. В исполнительном модуле при создании объекта класса текущего окна (при выполнении Settings5View::Settings5View()) создаётся объект смарт-пойнтера, в шаблон которого устанавливаются параметры текущего окна.
    4. В исполнительном модуле при создании объекта класса текущего окна описывается функция обработки обратного вызова.
    5. При открытии окна (Settings5View::setupScreen()) выполняется метод установки callback для нужного класса колеса прокрутки. В метод установки callback передаётся смарт-пойнтер на функцию обработки. Метод установки callback для события в колесе прокрутки может быть любым из имеющегося перечня методов класса колеса прокрутки. Методы описаны в API.⁰С