

UI Development Fundamentals

This presentation is available on support.touchgfx.com

This presentation focus on the fundamentals elements when creating UI application with TouchGFX.

Starting out by looking software architectural concepts, the is used within TouchGFX, then talking a bout the Designer and widgets, End finally looking at how to ingrate you own C++ code in an application.

This presentation primarily targets software developers with a basic skill-set within C++ and embedded GUI development on STM32.

Since this is a fundamentals video, newcomers to this filed, can also benefit from watching this presentation, but a basic understanding of what TouchGFX is, is advised.

This can for example be achieved by watching the presentations:

TouchGFX Introduction

(<https://support.touchgfx.com/docs/resources/presentations#touchgfx-introduction>), and

TouchGFX Technical Introduction

(<https://support.touchgfx.com/docs/resources/presentations#touchgfx-technical-introduction>)

To have a better general understanding of embedded graphics it can also be helpful to watch:

Embedded Graphics - Basic Concepts

(<https://support.touchgfx.com/docs/resources/presentations#embedded-graphics---basic-concepts>)

A step-by-step guide for getting started with TouchGFX is also available as the workshop:

UI Development - Getting Started

(<https://support.touchgfx.com/docs/resources/presentations#ui-development---getting-started>)

The main chapter covered in this presentation are

UI Development (<https://support.touchgfx.com/docs/development/ui-development/ui-development-introduction>)

This presentation takes approximately ~ 1 hour

Agenda

1 Introduction

2 Software architecture

3 Working with TouchGFX Designer

4 Designer user guide

5 UI components

6 Working with TouchGFX in User Code



Introduction

With this presentation, you get a fundamental knowledge about TouchGFX UI Development. You will learn:

- The basics of the software architecture
- Getting started working with the Designer and its UI components
- Discovering TouchGFX Engine features



Introduction

Further reading

- Go to the TouchGFX documentation site :

<http://support.touchgfx.com/>

- Slides in this workshop will refer to relevant documentation pages. Links will be in the lower corner of the slides

- A good place to start reading following this workshop is:

[UI Development Introduction](#)



TouchGFX Development

Main components:



Main activities:

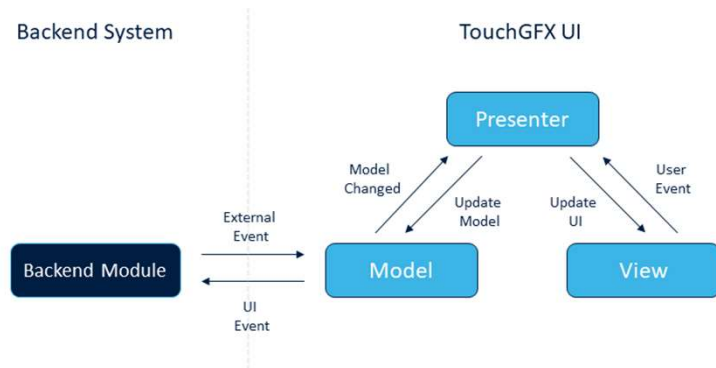


In this presentation, we will focus on the UI development activity

We go deeper into the different activities in the other presentations

Software architecture

- Model-View-Presenter (MVP)
 - View
 - Model
 - Presenter
- Benefits
 - Separation
 - Unit testing
 - Decoupled code



[Documentation Link: Model-View-Presenter Design Pattern](#)

7

Model-View-Presenter Design Pattern

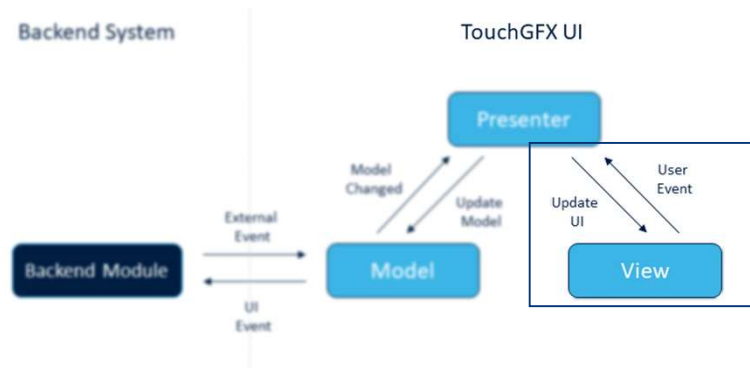
View: Handles the ui elements

Presenter: Hnadles the buisness logic behind the UI elements

Model: Stores information through out the application and handle communication with the rest of the system

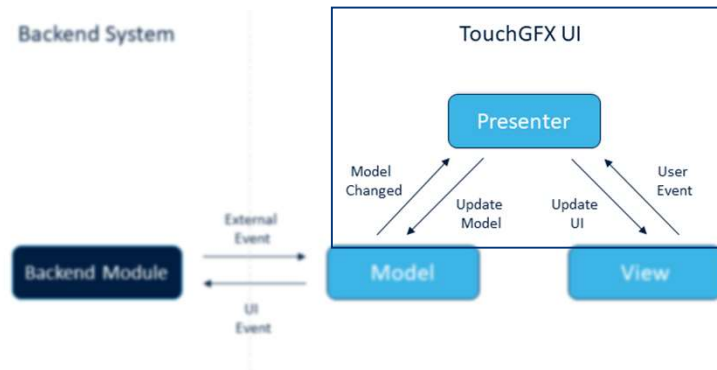
Software architecture

- View
 - Creation of UI elements
 - UI element manipulation
 - “Reception” of user input
 - Sending of user input to Presenter
 - Update of UI based on data from Presenter



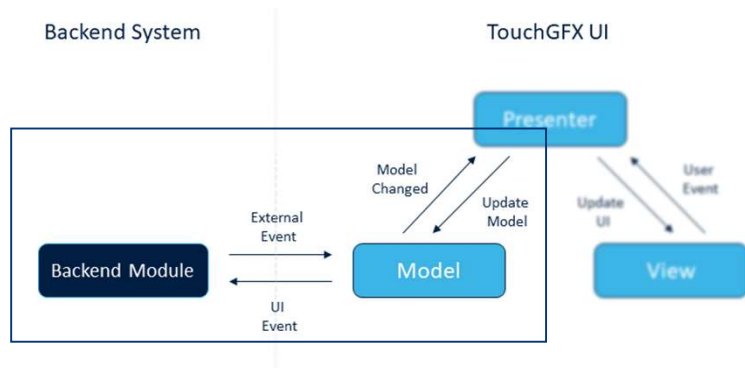
Software architecture

- Presenter
 - Mediator between Model and View
 - Receives data from the Model and prepares it for the View
 - Receives input from the View and sends it to the Model
 - Business logic for UI
 - No knowledge about UI implementation



Software architecture

- Model
 - Communicates with the backend
 - Stores data
 - Passes data to the Presenter



Software architecture

- MVP – Example

- When update is pressed, a get time request is sent to the Model via the Presenter
- The Model sends time data to the Presenter
- The Presenter converts the data into something the Clock widget can use and sends it to the View
- The View updates the Clocks with the new data

The image shows two windows. The top window is a terminal titled 'build-bin\multitester.exe' showing the following log output:

```
View
Request time update
Presenter
Request new Time String
Model
Time String:
Thu Oct 22 13:39:20 2020
Presenter
Time String to int conversion:
Hours: 13, Minutes: 39, Seconds 20
View
UI update with time ints
```

The bottom window is titled 'MVP Example' and displays a user interface with the time '13:39:20' and a clock widget. A blue button labeled 'Update Clock' is visible at the bottom left of the UI.

Software architecture

- The Screen Concept
 - Application consists of a number of “Screens”
 - Logical grouping of UI and UI logic
 - One View class and one Presenter class
 - One active screen at the time
 - Minimizes the amount of RAM-memory

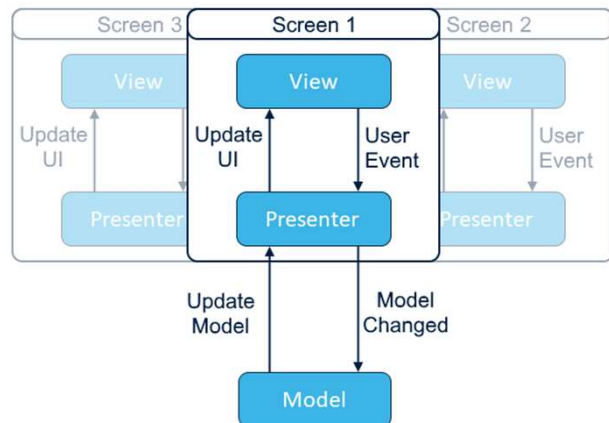


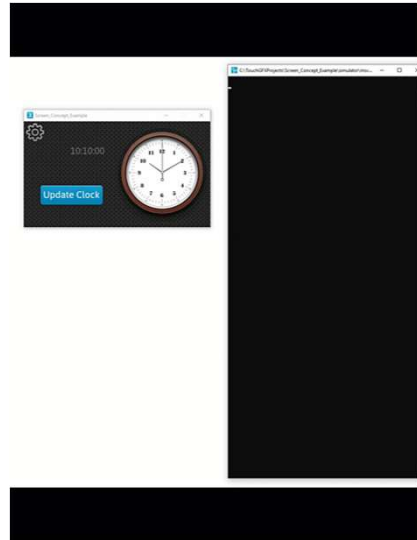
Illustration of the screen concept with screen 1 as the active screen

[Documentation Link: The Screen Concept](#)

12

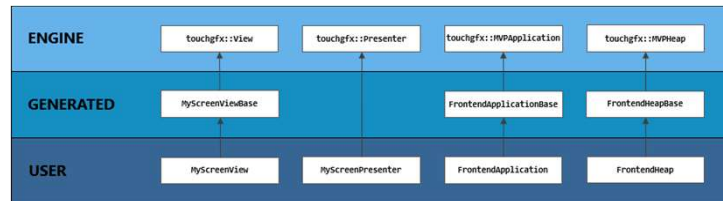
Software architecture

- Screen Concept: Example
 - When selecting clock mode, in the “settings” screen, the selected mode is stored in the model
 - When pressing update, the Clock also retrieves the clock mode from the model



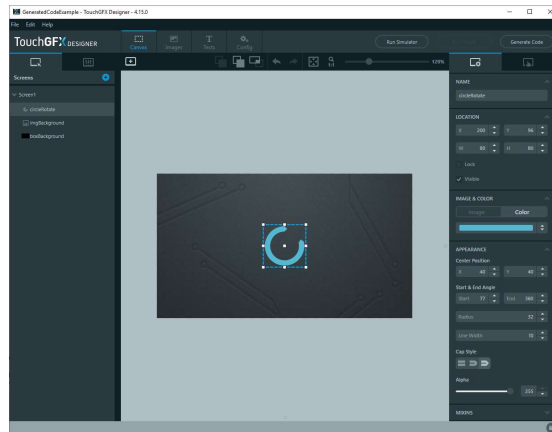
Software architecture

- Generated Code vs. User Code
 - Engine
 - Provided by TouchGFX
 - Generated
 - Generated by the Designer multiple times
 - Code for the widgets added in the Designer
 - Regenerated often
 - User
 - Generated by the Designer once
 - Created for handwritten code



Software architecture

- Rotating Circle Example: The Designer



life.augmented

[Documentation Link: Code Structure](#)

15

This example shows the relations between Generated Code and User Code. In the example we create an application with a rotating circle.

In this slide the circle is added to the application through the designer.

Software architecture

- Rotating Circle Example: The generated code

```
1 //***** THIS FILE IS GENERATED BY TOUCHFX DESIGNER, DO NOT MODIFY *****//
2 //***** THIS FILE IS GENERATED BY TOUCHFX DESIGNER, DO NOT MODIFY *****//
3
4 #include <gui_generated/screen1_screen/ScreenViewBase.hpp>
5 #include <touchfx/Color.hpp>
6 #include "BitmapBase.hpp"
7
8 ScreenViewBase::ScreenViewBase()
9 {
10
11     touchfx::CanvasWidgetRenderer::setupBuffer(canvasBuffer, CANVAS_BUFFER_SIZE);
12
13     _background.setPosition(0, 0, 480, 272);
14     _background.setColor(touchfx::Color::getColorFrom24BitRGB(0, 0, 0));
15
16     imgBackground.setXY(0, 0);
17     imgBackground.setBitmap(touchfx::Bitmap(BITMAP_B6_ID));
18
19     circleRotate.setPosition(200, 96, 88, 80);
20     circleRotate.setCenter(40, 40);
21     circleRotate.setRadius(32);
22     circleRotate.setLineWidth(10);
23     circleRotate.setArc(77, 360);
24     circleRotate.setCapPrecision(80);
25     circleRotatePainter.setColor(touchfx::Color::getColorFrom24BitRGB(85, 185, 200));
26     circleRotate.setPainter(circleRotatePainter);
27
28     add(_background);
29     add(imgBackground);
30     add(circleRotate);
31 }
32
33 void ScreenViewBase::setUpScreen()
34 {
35 }
36 }
```



life.augmented

[Documentation Link: Code Structure](#)

16

In this slide we can see the base view for the rotating circle example. This is the ScreenBaseView.cpp, in this file the code for adding and setting up the circle, that we added in in designer are generated. Since this file consists of generated code, the file is a read-only file, and are regenerated when you select to run the demo on either simulator or target, or generate code is selected.

- Rotating Circle Example: The User Code

```
1 #include <gui/screen_screen/ScreenView.hpp>
2
3 ScreenView::ScreenView()
4 {
5 }
6
7 void ScreenView::setupScreen()
8 {
9 }
10
11 void ScreenView::tearDownScreen()
12 {
13 }
14
15 void ScreenView::handleTickEvent()
16 {
17     // sets the new angle to rotate circleRotate //
18     circleRotate.invalidate();
19
20     circleRotate.setArc(circleRotate.getArcStart() + 1, circleRotate.getArcEnd() + 1);
21
22     circleRotate.invalidate();
23 }
```

In this slide we can see the view for the rotating circle example.

This is the ScreenView.cpp, which are designed to be where user code should be inserted.

Since the ScreenView are inherited from the ScreenBaseView, we are able to access the widgets that have been added in the designer and generated code for in the ScreenBaseView.

Software architecture

- Rotating Circle Example: The Demo



[Documentation Link: Code Structure](#)

18

This slide shows the demo running in the Simulator

Touchgfx designer is the main tool used in the UI development activity. With this tool users will setup, design and create the looks of the project

A Widget is an abstract definition of something that can be drawn on the screen and be interacted with

- Numerous standard widgets available
 - Customizable from Designer and Code
 - The adding order determines the display order
- Possible to create custom widgets



A Button widget with an Image widget as a background



life.augmented

[Documentation Link: Widgets](#)

20

A Widget is an abstract definition of something that can be drawn on the screen and be interacted with. Multiple widgets, like buttons, images or graphs are available.

Users can add widgets to their UI and customize them with the supplied properties. The order in which they are added determines the order in which they are displayed.

The TouchGFX generated code for the widgets can be used as a source of inspiration or as base by users to create custom widgets.

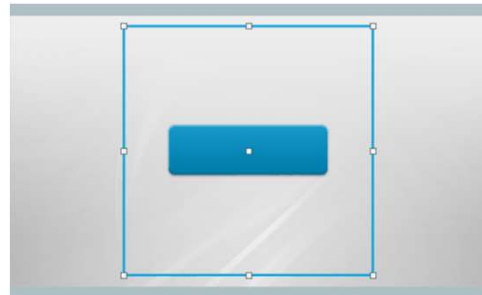
Custom Widget documentation link :

<https://support.touchgfx.com/docs/development/ui-development/touchgfx-engine-features/custom-widgets/>

Containers

A Container is a component that contains child nodes

- Children can be widgets or other containers
- Position of widgets defined relatively to the parents
- Widgets are added to a container by dragging widgets into the container in the tree view
- Containers act as viewports
 - Only the parts of the children that intersect with the geometry of the container are visible



A container acting as a view port



life.augmented

[Documentation Link: Containers](#)

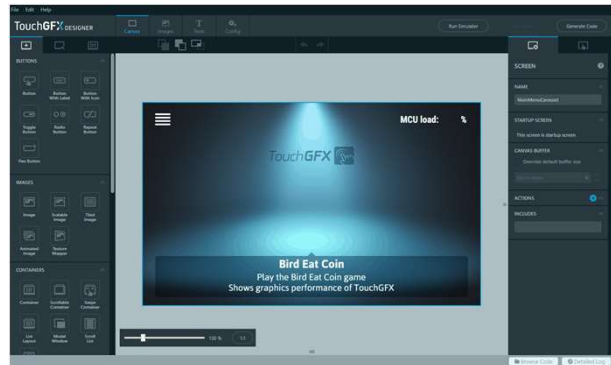
21

Containers are components containing child nodes, which can be widgets or other containers. Within TouchGFX Designer, the childs are added to a container by dragging them within the container in the tree view. Containers can be used for many things, for example moving multiple elements at a time since the positions of the child elements are relative to the parent container. As shown in the gif, elements within a container will be displayed if their coordinates match its display window.

Simulator

TouchGFX includes a simulator for UI testing

- Building and running on target can be time-consuming
- You can run the simulator in 3 ways by use of:
 - TouchGFX Designer
 - TouchGFX Environment
 - Visual Studio
- Option to create User code that only runs when using the simulator



Launching the simulator from TouchGFX Designer



[Documentation Link: Simulator](#)

22

TouchGFX includes a simulator to replicate the running UI prototype instead of flashing on target a project every time. This can also be used for projects without hardware configured yet.

There are two ways to run the simulator:

Using TouchGFX Designer

Using TouchGFX Environment

For testings, It is possible to create User code that will be only ran when using the simulator

Compiling

A TouchGFX application can be compiled for the PC simulator or for the target hardware

- PC simulator: Projects compiled using GCC or Visual Studio
- Hardware target: Projects compiled with the active toolchain
 - Active toolchain set in CubeMX



life.augmented

[Documentation Link: Compiling](#)

23

Projects are compiled using GCC or Visual studio, or using the toolchain of your choice (Keil μ Vision, IAR Embedded Workbench, STM32CubeIDE). Be aware that the initial active toolchain is STM32CubeIDE. Another toolchain can be set within the CubeMX tool.

Using IDEs with TouchGFX documentation link :

<https://support.touchgfx.com/docs/4.14/development/ui-development/working-with-touchgfx/using-ides-with-touchgfx>

Flashing

Flashing projects on STM32-based boards can be done with different toolchains.

- Building generates a binary (.hex or .elf)
 - Flash with STlink or STM32Cube Programmer
- Running on target is done using GCC
 - Command can be overridden in configurations



life. augmented

[Documentation Link: Flashing](#)

24

Building a project generates a binary file which is flashed on target using STM32CubeProgrammer or STlink for older projects
GCC is used as default to Run a project on target from TouchGFX Designer. The generating and running commands can be overwritten by users.
Download link for CubeProgrammer and ST-LINK Utility:
<https://support.touchgfx.com/docs/introduction/installation#installing-stm32cubeprogrammer>

Debugging

A TouchGFX project is a set of C++ files generated by TouchGFX Designer, TouchGFX Generator and written by developers

- Can be debugged as any other C++ application
- Target debugging with an IDE
 - Performance testing
- Simulator Debugging
 - Faster process than on target
- Using the DebugPrinter
 - Prints debug messages on the display



life.augmented

[Documentation Link: Debugging](#)

25

A TouchGFX project is a C++ project, meaning it can be debugged as any regular C++ project.

Debugging by running on target is useful for performance testing such as the animation speed, update frequency and responsiveness of the UI elements.

Debugging with the simulator is much faster and more efficient for testing the look and logic of the UI.

The DebugPrinter option will print information on the display.

Getting started

Using UI templates is a good way to start a project or to understand how to work with widgets and TouchGFX

- Widget showcasing
- Complete demos
- Can be used as source of inspiration

Online applications are out-of-the box applications dedicated to an ST evaluation kit

- More complex than UI templates
- Explore different features of the TouchGFX framework
- Can include sample integration with hardware



[Documentation Link: Examples](#)

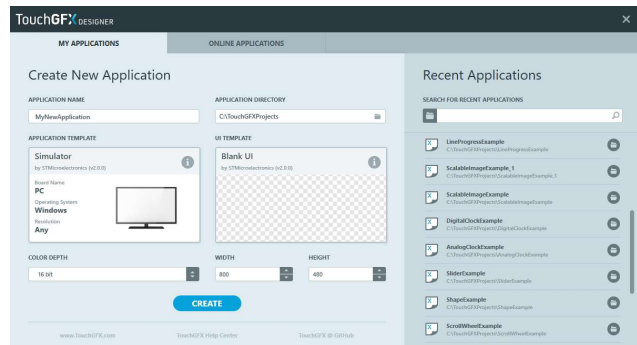
26

TouchGFX Designer offers multiple UI templates which showcases the functionalities of widgets and the options of Designer. The code for those examples and demos can be used as a source of inspiration to get started with a custom project. Some ST evaluation kits also have dedicated demos, called Online application which can explore advanced concepts and features.

Startup Window

This is where new projects can be configured and created

- Possibility to work with the Simulator or with an Application or UI template



Startup window when opening TouchGFX Designer



This is where new projects can be configured and created:

- Application name
- Application directory path
- Width, Height and Color depth

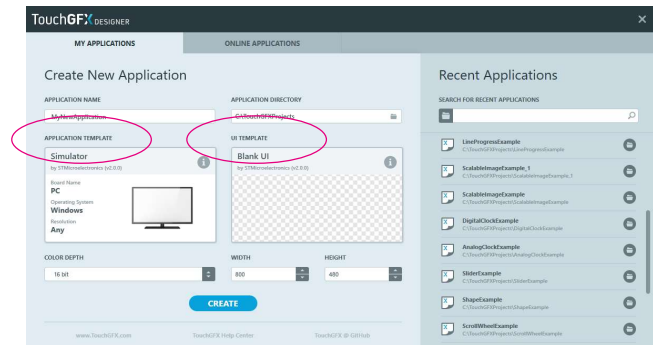
Application templates for ST evaluation kits and UI templates can be selected.
Online applications directly combine an AT and a UI demo
Perfect for understanding how to work with TouchGFX Designer
This can be a useful source of inspiration for user code

Online applications are out-of-the box applications for specific hardware solutions

Startup Window

This is where new projects can be configured and created

- Possibility to work with the Simulator or with an Application or UI template



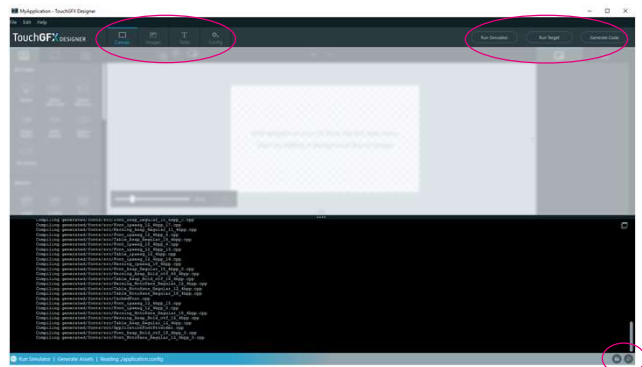
Startup window when opening TouchGFX Designer



Main Window

Contains a Navigation Bar, Command Buttons, Notification Bar, and a Detailed log

- Navigation bar
 - *Canvas*: drag and drop application building
 - *Images*: management of images used
 - *Text*: management of text and typographies
 - *Config*: configuration of project settings



The main window options



life.augmented

[Documentation Link: Main Window](#)

30

After creating a project from the startup window, users can start developing the UI. But lets first go through the tabs and options. The Main Window consists of a Navigation Bar, Command Buttons, Notification Bar, and a Detailed log

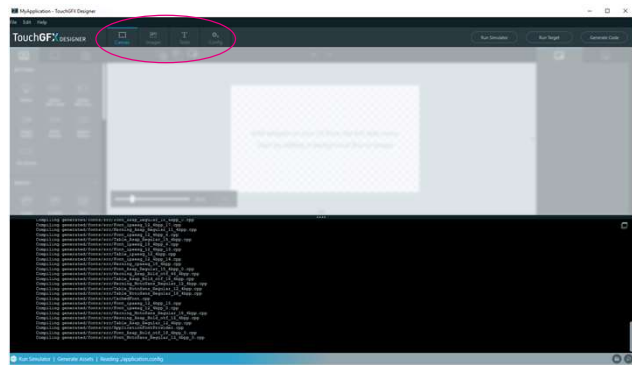
In TouchGFX Designer, the navigation is done through the Navigation Bar. It consists of:

- Canvas* for drag and drop application building
- Images* for management of images used
- Text* for management of text and typographies
- Config* for configuration of various project settings

Main Window

Contains a Navigation Bar, Command Buttons, Notification Bar, and a Detailed log

- Navigation bar
 - *Canvas*: drag and drop application building
 - *Images*: management of images used
 - *Text*: management of text and typographies
 - *Config*: configuration of project settings



The main window options



[Documentation Link: Main Window](#)

31

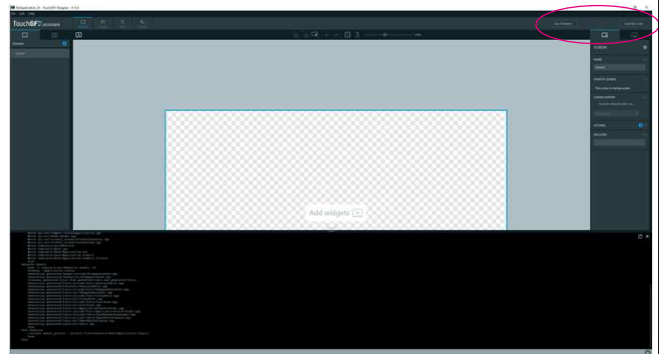
The Main Window consists of a Navigation Bar, Command Buttons, Notification Bar, and a Detailed log

In TouchGFX Designer, the navigation is done through the Navigation Bar. It consists of:

- Canvas* for drag and drop application building
- Images* for management of images used
- Text* for management of text and typographies
- Config* for configuration of various project settings

Main Window

- Project can be run on target or on PC simulator
 - Both commands trigger assets, code generation and compiling
 - Key shortcuts are respectively “F5” and “F6”
- Notification bar and Detailed log are at the bottom of the Main Window
 - Shows status of running command
 - Opens log of last command
 - Can be toggled with keys “Alt” + “L”



Running and generating code commands, Navigation bar and Browser Code positions



life.augmented

[Documentation Link: Main Window](#)

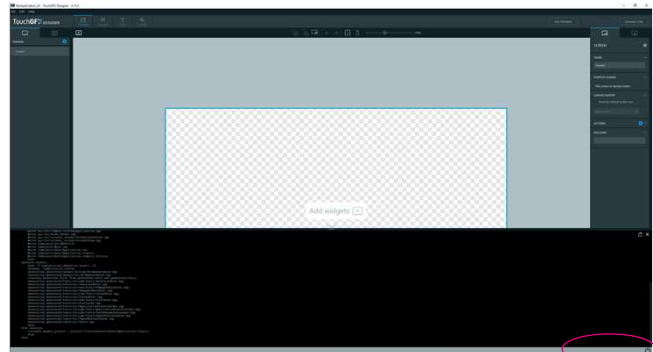
32

The code for the UI can be generated by pressing the command "generate code" or with the key F7. From Designer it is possible to run the simulator or flash projects on the boards using the dedicated buttons or by pressing respectively F5 or F6.

The status of a running command is shown in the notification bar. Clicking on it opens a detailed log to have more information on what went right or wrong.

Main Window

- Project can be run on target or on PC simulator
 - Both commands trigger assets, code generation and compiling
 - Key shortcuts are respectively “F5” and “F6”
- Notification bar and Detailed log are at the bottom of the Main Window
 - Shows status of running command
 - Opens log of last command
 - Can be toggled with keys “Alt” + “L”



Running and generating code commands, Navigation bar and Browser Code positions



life.augmented

[Documentation Link: Main Window](#)

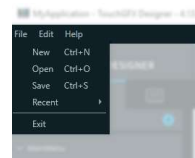
33

Pressing on the Notification Bar opens a window showing the full log of the last command. It can also be toggled with Alt + L

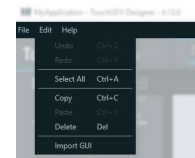
File Menu

Consists of a File, Edit and Help menu items

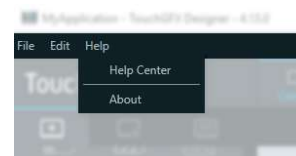
- File Menu used for creating/opening/saving projects
- Edit Menu interacts with UI elements. It can be used to import UI templates or demos after project creation
- Help Menu redirects to official support forum. Also contains Software License Agreement



File menu item in File Menu



Edit menu item in File Menu



Help menu item in File Menu



life.augmented

Documentation Link: [File Menu](#)

34

The file menu is made of three elements/ File, edit and help menu items. The File Menu is used for creating, editing and saving projects. A link to the official forum is also given if in need of support or to ask questions.

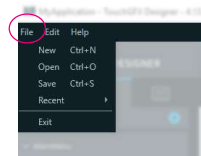
Link to the forum “Community” :

<https://community.st.com/s/topic/0TO0X0000003iw6WAA/touchgfx>

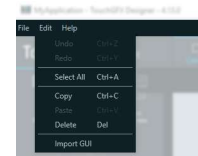
File Menu

Consists of a File, Edit and Help menu items

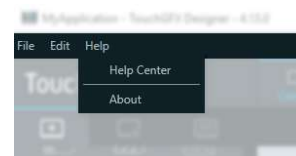
- File Menu used for creating/opening/saving projects
- Edit Menu interacts with UI elements. It can be used to import UI templates or demos after project creation
- Help Menu redirects to official support forum. Also contains Software License Agreement



File menu item in File Menu



Edit menu item in File Menu



Help menu item in File Menu



life.augmented

[Documentation Link: File Menu](#)

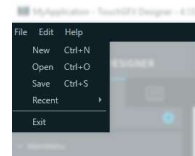
35

File Menu is used for creating/opening/saving projects and exiting Designer

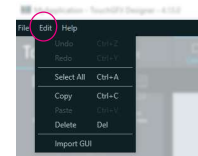
File Menu

Consists of a File, Edit and Help menu items

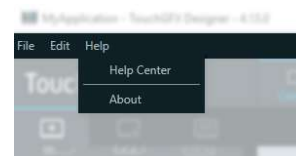
- File Menu used for creating/opening/saving projects
- Edit Menu interacts with UI elements. It can be used to import UI templates or demos after project creation
- Help Menu redirects to official support forum. Also contains Software License Agreement



File menu item in File Menu



Edit menu item in File Menu



Help menu item in File Menu



life.ougmented

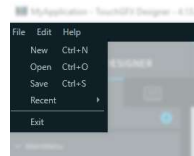
[Documentation Link: File Menu](#)

36

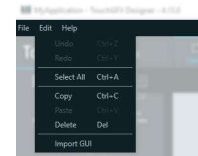
File Menu

Consists of a File, Edit and Help menu items

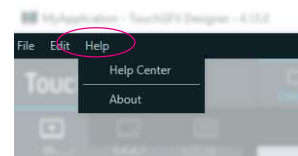
- File Menu used for creating/opening/saving projects
- Edit Menu interacts with UI elements. It can be used to import UI templates or demos after project creation
- Help Menu redirects to official support forum. Also contains Software License Agreement



File menu item in File Menu



Edit menu item in File Menu



Help menu item in File Menu



life.ougmented

[Documentation Link: File Menu](#)

37

Images View

Used to manage images used in a TouchGFX Project

- Images are located under the assets\images folder in the project folder

Contains 3 sections: the tree view on the left, the table view in the middle and the properties view on the right

- *Tree view*: overview of the images and folders
- *Table view*: list of images located in the selected folder
- *Properties view*: properties of an image

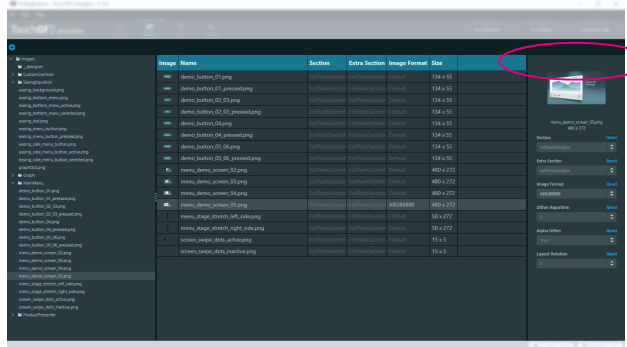


Image View of the UI template "Demo 1"

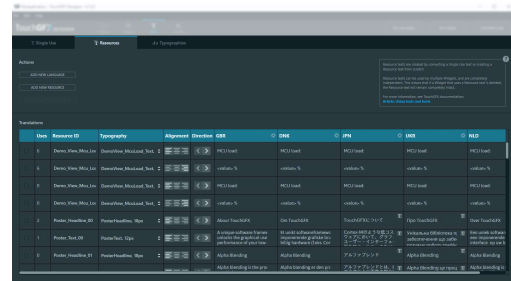


The properties view is where users can edit the properties of an image, like the bit per pixel format or where they are stored in the board.

Texts View

Used for configuration of texts, translations and typographies

- Single Use and Resources tabs both contain an overview of texts but have different usage



Text View of the UI template "Demo 1"

Working with texts and the Text Views is explained in the ["Texts and Fonts"](#) article



[Documentation Link: Texts View](#)

42

The text view is where users can handle all the text used in the UI. It is where the fonts and typographies can be edited. Using texts can be complicated. This section is explained in detail in the related articles in the documentation. Texts and Fonts documentation link :

<https://support.touchgfx.com/docs/development/ui-development/touchgfx-engine-features/texts-and-fonts>

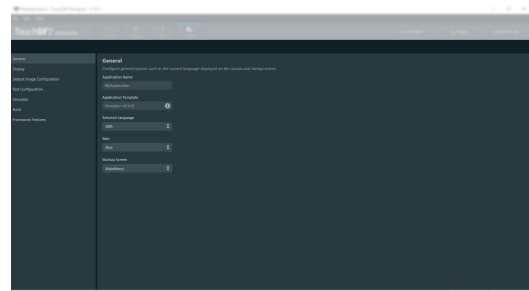
Languages and Characters link :

<https://support.touchgfx.com/docs/development/ui-development/touchgfx-engine-features/languages-and-characters>

Config View

Various settings affecting the project can be configured in the Config View

- General settings
 - Application name or startup screen
- Display setting
 - Dimensions, display orientation and color depth
- Default image settings
 - Settings of all images in the project, unless overwritten in the Images View



Config View



life.augmented

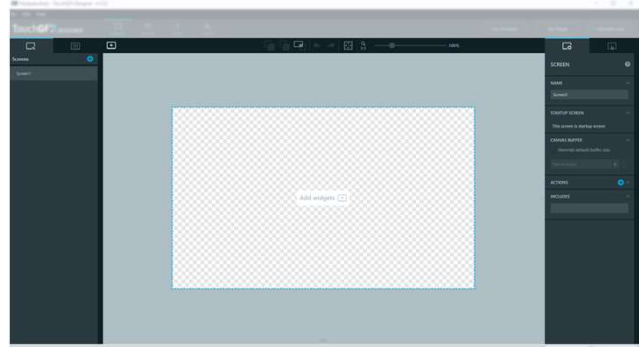
[Documentation Link: Config View](#)

43

The config view is where users can configure advanced settings for the project, for example overwriting the compiling and flashing commands.

Canvas View

- Landing screen after project creation
 - Visual representation of the selected screen or custom container
- The left bar is where screens and custom containers are configured
 - Additional Screens/Custom containers added by pressing the blue icon
 - Screen and Widget order changed by dragging



The Canvas View



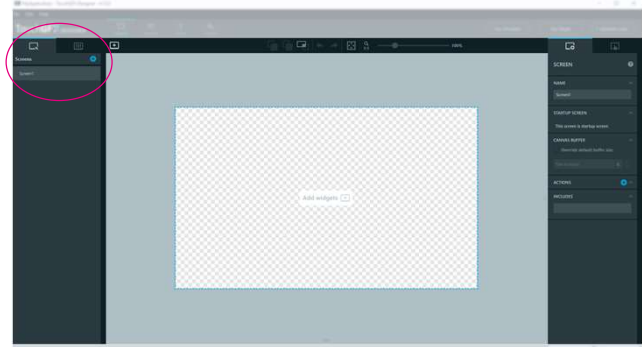
[Documentation Link: Canvas View](#)

44

Screens and Widgets order changed by dragging them below or above others in the tree


Canvas View

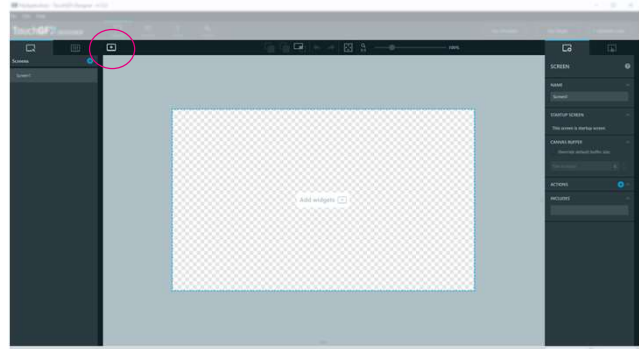
- Landing screen after project creation
 - Visual representation of the selected screen or custom container
- The left bar is where screens and custom containers are configured
 - Additional Screens/Custom containers added by pressing the blue icon
 - Screen and Widget order changed by dragging



The Canvas View

Canvas View

- “Add widget” button  opens the widget menu
 - Contains all available widgets grouped by categories
 - Clicking a widget will add it to the canvas
- The right bar contains two tabs: Properties and Interaction
 - Properties is where widgets and screen properties are set
 - Interaction is where widgets and screen interactions can be created and defined



The Canvas View




[Documentation Link: Canvas View](#)

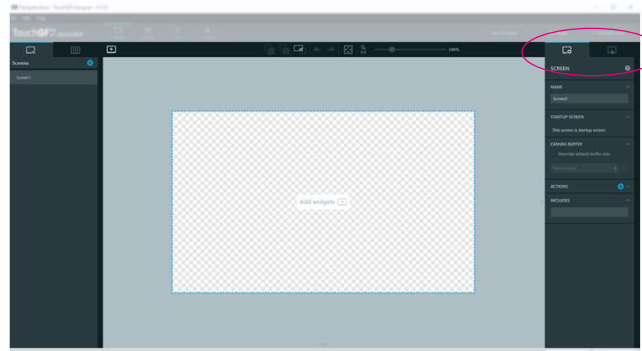
46

The add button widget will open the widget menu, where users can select which widget to add to their UI. Those widgets will be displayed in the canvas view.

The size and position of the widgets can be manipulated by users within the center section, representing the looks of the selected screen.

Canvas View

- “Add widget” button  opens the widget menu
 - Contains all available widgets grouped by categories
 - Clicking a widget will add it to the canvas
- The right bar contains two tabs: Properties and Interaction
 - Properties is where widgets and screen properties are set
 - Interaction is where widgets and screen interactions can be created and defined



The Canvas View



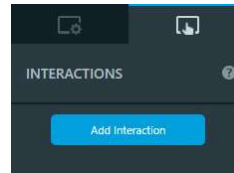
[Documentation Link: Canvas View](#)

47

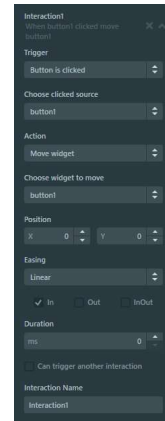
The right bar contains the properties and interaction tabs. This is where the properties of screens and widgets are set, like their position and width and height. The interaction tab is used to define widgets and screen interactions, for example a fading in image when entering the main screen at startup

Interactions

- An Interaction consists of a trigger and an action
 - A Trigger is what will start the interaction
 - An Action is what will happen after a Trigger has been emitted
 - The number of triggers and actions available depends on the widgets added to the project
 - Interactions can be chained



Interaction tab in TouchGFX Designer



Moving a widget when a button is pressed



life.augmented

[Documentation Link: Interactions](#)

48

Interactions are often the heart of an UI. The interactions set within this tab in TouchGFX designer will generate all the code necessary to have them running. Users can add or modify the logic of the interactions through user code to add more complex animations and interactions.

Widgets

- Wide range of widgets available
- Categories based on properties and usage
 - Custom containers will appear as a new category
- Custom widgets cannot be seen nor configured from Designer



Widget tab

[Documentation Link: Custom Widgets](#)

50



life.ougmented

The widget menu orders the widgets available in TouchGFX Designer by categories based on their properties and usage; for example the different types of button widgets are under a category called Buttons. Additional widgets maybe added with futur releases. Clicking on an icon will add a widget with a name and initial settings to the selected screen.

Creating Custom containers will spawn a new category at the bottom of the widget menu, where they will made available. They can be used different scenarions and are useful for advanced UIs. Some widgets like scroll list or scroll list needs custom containers to function.

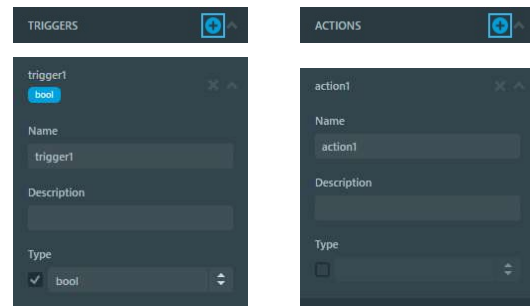
It is possible for users to create widgets through code for specific needs in a project. However they will not be shown nor be configured in the canvas view of TouchGFX Designer, but will be displayed in the simulator or on the board.

The button with label widget it is a widget aware of touch events. It is used as a button, sending a callback when released. It has two states: pressed and released each associated with an image and text. The images should not look the same, but need to be of the same size, in order to graphically show the user if the widget is pressed. As it does not involve complex animations or calculations it is considered a light widget with good performances on most MCUs.

Custom containers can be used for a wide range of application. The coordinates of child elements are related to the container's coordinates, it is then useful for moving multiple elements at once. It is also useful when having a recurrent group of elements in different screens, like a menu bar consisting of three buttons with boxes with text areas as shown on this slide. Depending on the widgets used within the container the performances can vary.

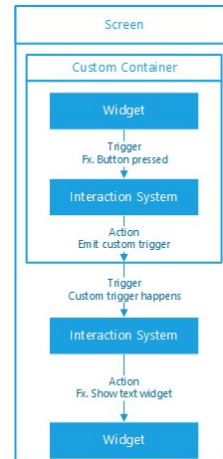
Working with TouchGFX in User Code

- Actions and Triggers
 - Custom interactions
 - Interface between Designer and User Code
 - Created under screen properties
 - After creation, then added to the interaction system



Working with TouchGFX in User Code

- Triggers
 - Enable a Screen to react on events from a Custom Container
 - Custom Container can emit a custom Trigger as an Action
 - Enable a Screen to react to a custom Trigger from a Custom Container
 - Able to pass data from Custom Container to Screen
 - Generated as C++ callback
 - After creation, the trigger is added to the interaction system
 - Can be emitted from User Code



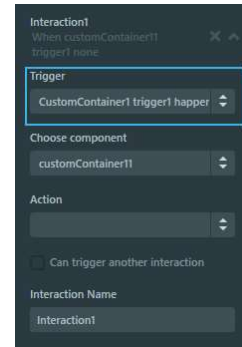
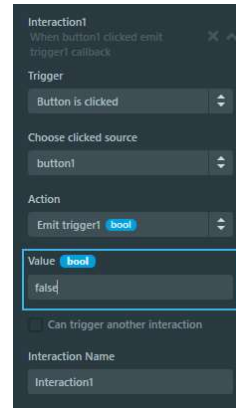
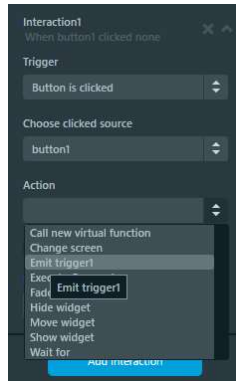
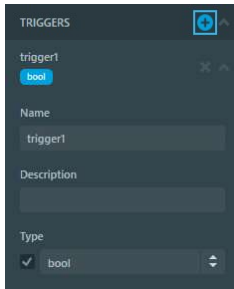
Flow for a Custom trigger between a widget in a custom container and a widget in a screen

[Documentation Link: Custom Triggers and Actions](#)

55

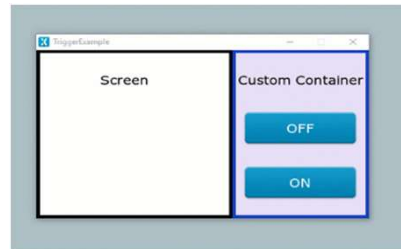
Working with TouchGFX in User Code

- Creating and using a Trigger



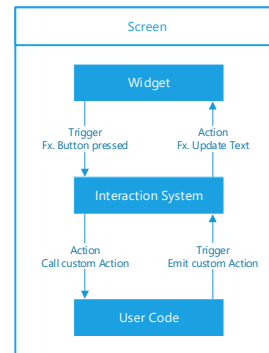
Working with TouchGFX in User Code

- Triggers – Example
 - Custom Container added to a Screen
 - Pressing ON or OFF emits a ON or OFF Custom Trigger
 - Based on the Custom Trigger, the ON or OFF text will show



Working with TouchGFX in User Code

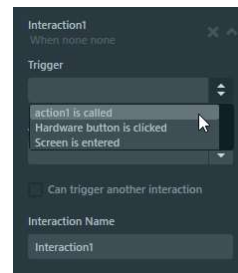
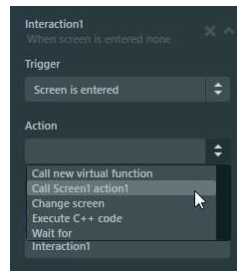
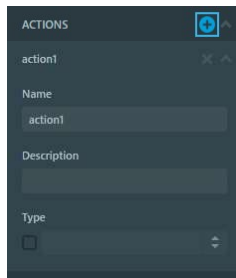
- Actions
 - Interface between Screen/Custom Container and User Code
 - The Interaction System can call Custom Actions in User Code as Actions
 - User code can emit Custom Actions as Triggers via the Interaction System
 - Able to pass data from Screen or Custom Container to User Code
 - Generated as a virtual C++ methods
 - The action is added to the interaction system



Relations between Widgets, Interactions System and User Code, when using Custom Actions

Working with TouchGFX in User Code

- Actions: Designer Setup Example



Working with TouchGFX in User Code

- Actions: User Code Example

```
ScreenView.cpp
1 #include <gui/screen_screen/ScreenView.hpp>
2
3 ScreenView::ScreenView()
4 {
5 }
6
7
8 void ScreenView::setupScreen()
9 {
10     ScreenViewBase::setupScreen();
11     action1();
12 }
13
```

```
ScreenView.hpp
1 #ifndef SCREENVIEW_HPP
2 #define SCREENVIEW_HPP
3
4 #include <gui_generated/screen_screen/ScreenViewBase.hpp>
5 #include <gui/screen_screen/ScreenIPresenter.hpp>
6
7 class ScreenView : public ScreenViewBase
8 {
9 public:
10     ScreenView();
11     virtual ~ScreenView() {}
12     virtual void setupScreen();
13     virtual void tearDownScreen();
14
15     virtual void action1();
16 protected:
17 };
18
19 #endif // SCREENVIEW_HPP
20
```

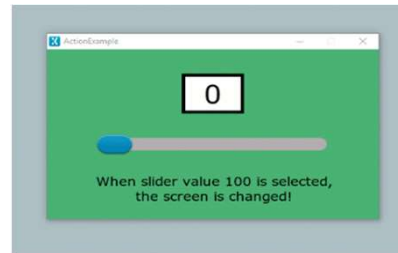
```
ScreenView.action1
1 #include <gui/screen_screen/ScreenView.hpp>
2
3 ScreenView::ScreenView()
4 {
5 }
6
7
8 void ScreenView::setupScreen()
9 {
10     ScreenViewBase::setupScreen();
11 }
12
13 void ScreenView::tearDownScreen()
14 {
15     ScreenViewBase::tearDownScreen();
16 }
17
18 void ScreenView::action1()
19 {
20 }
21
22
```



life.ougmented

Working with TouchGFX in User Code

- Actions: Demo
 - When the slider is dragged, the value of the slider is passed with an Action to User Code, which updates the text in the box
 - When the finger (mouse) releases the slider, another action sends the confirmed value to the User Code via an Action
 - If the value is 100 a Custom Action is sent from user code, to changes screen



Working with TouchGFX in User Code

- Mixins
 - Extend Widgets functionality
 - Mixin functionalities are used in User Code
 - Four different Mixins
 - Move Animator
 - Fade Animator
 - Draggable
 - Click Listener
 - Can be added to a widget in the Designer
 - Container Widgets cannot use Fade Animator

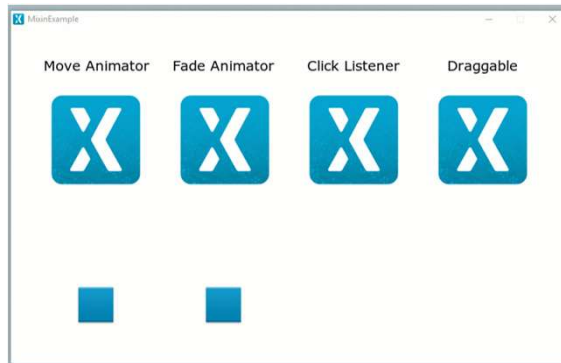


```
touchgfx::MoveAnimator< touchgfx::Box > box;
```

*The Mixins properties in the TouchGFX Designer (top)
The Mixin Move Animator added to a Widget in User
Code (bottom)*

Working with TouchGFX in User Code

- Mixin – Example
 - The four images, has the four Mixins added to them, denoted by the text above them.



User Code

- Everything done in the Designer, can also be done in User Code
 - But the Designer can help you with a lot of things
- Avoid going back and forth between User Code and Generated Code
- Remember to utilize the MVP Pattern
- Inspect Generated code
- Reuse code from examples
- [The TocuGFX API: Button](#)



- Generated Code from Animated Image Example

```
MainWindowBase::MainWindowBase() :
    buttonCallback(this, @MainWindowBase::buttonCallbackHandler),
    animationEndedCallback(this, @MainWindowBase::animationEndedCallbackHandler)
{
    __background.setPosition(0, 0, 480, 272);
    __background.setColor(touchgfx::Color::getColorFrom24bitRGB(0, 0, 0));

    boxBackground.setPosition(0, 0, 800, 480);
    boxBackground.setVisible(false);
    boxBackground.setColor(touchgfx::Color::getColorFrom24bitRGB(0, 0, 0));

    imgBackground.setXY(0, 0);
    imgBackground.setBitmap(touchgfx::Bitmap(BITMAP_BG_ID));

    animation.setXY(161, 18);
    animation.setBitmaps(BITMAP_ANI_01_ID, BITMAP_ANI_14_ID);
    animation.setUpdateFlcksInterval(2);
    animation.setDoneAction(animationEndedCallback);

    btnToggle.setXY(175, 195);
    btnToggle.setBitmaps(touchgfx::Bitmap(BITMAP_BTN_ID), touchgfx::Bitmap(BITMAP_BTN_PRESSED_ID));
    btnToggle.setLabelText(touchgfx::TypedText(T_TEXTSTART));
    btnToggle.setLabelColor(touchgfx::Color::getColorFrom24bitRGB(213, 115, 0));
    btnToggle.setLabelColorPressed(touchgfx::Color::getColorFrom24bitRGB(213, 115, 0));
    btnToggle.setAction(buttonCallback);

    add(__background);
    add(boxBackground);
    add(imgBackground);
    add(animation);
    add(btnToggle);
}
```

```
void MainWindowBase::buttonCallbackHandler(const touchgfx::AbstractButton& src)
{
    if (&src == &btnToggle)
    {
        //buttonClicked
        //When btnToggle clicked execute C++ code
        //Execute C++ code
        if (animation.IsAnimatedImageRunning())
        {
            animation.pauseAnimation();
            btnToggle.setLabelText(TypedText(T_TEXTSTART));
        }
        else
        {
            animation.startAnimation(animation.isReverse(), false, true);
            btnToggle.setLabelText(TypedText(T_TEXTSTOP));
        }
    }
}
```



Here we see relation between the code that is generated and user code, showing how to interact between user code and generated code. The two screen shots are snippets of code from the base view, to the left, and view to the right. The code are from the Animated Image example and can be accessed from the TouchGFX Designer.

Where to go next?

- More information can be found in the UI development section:
[UI Development](#)
- Get started on UI developing with the Hands-on Workshop:
[UI Development - Getting Started](#)
- Or by following our tutorials:
[Tutorials](#)

Thank you

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to www.st.com/trademarks.

All other product or service names are the property of their respective owners.



For further support and content,
you can visit the **TouchGFX** online
documentation site
<https://support.touchgfx.com>

