

TouchGFX Workshop

UI development – Getting Started

Agenda

Creating your own UI application

4









Goal of this workshop

- Get up and running so that you can do UI development on you own
 - Installation of required tools
 - Running an example using the Simulator
 - Flashing your STM32 Evaluation Kits (if available)
 - Creating your own TouchGFX UI application



Further reading

- You will find a lot of help afterwards in the TouchGFX documentation site: <u>https://support.touchgfx.com/</u>
- Slides in this workshop will refer to relevant documentation pages. Links will be in the lower right corner of the slides
- A good place to start reading following this workshop is: <u>Getting started - what's next</u>



TouchGFX Development

• Main Components:



• Main Activities:

- In this workshop we will focus on the component "TouchGFX UI Application" and thus the activity "UI Development"
- For the other components we will select a premade Application Template for an STM32 Evaluation Kit





- In this workshop we will focus on TouchGFX UI development so we will need the following tools
 - TouchGFX Designer
 - An IDE or text editor, like:
 - ST CubeIDE
 - Visual Studio
 - IAR Workbench
 - Keil uVision
 - Visual Studio Code
 - Notepad++
 - Emacs
 - ...
 - STM32 Cube Programmer (for flashing STM32 Evaluation Board if available)



- As we do not focus on the other Main Activities of TouchGFX development, we do not need to use the following tools.
 - STM32 CubeMX
 - TouchGFX Generator
- These should however be installed if you want to setup, modify or even create your project from scratch.
- Please note that if you want to change from the default compiler selection in the Application Template you need to install CubeMX.



- It is assumed that you all have the necessary tools installed for this workshop
- If not, please do so by following the instruction found here: Installation





Step GS-1: Create and run an application on your PC based on an example

- 1. Open TouchGFX Designer (or click "File/New" if already open)
- 2. Select one of the available examples by selecting a UI Templates
- 3. Keep the "Simulator" selection for the Application Template
- 4. Click the "Create" button

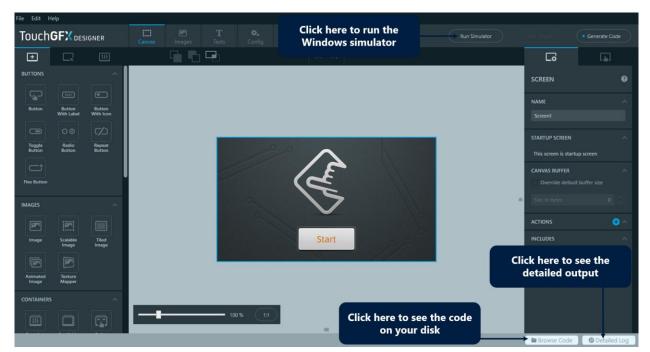
MY APPLICATIONS	ONLINE APPLICAT	IONS				
Create New Applicatio	n					
APPLICATION NAME	APPLICATION DIRE	CTORY				
MyApplication	C:\TouchGFXProj	ects 🗎				
APPLICATION TEMPLATE	UI TEMPLATE					
Simulator by STMicroelectronics (v2.0.0)	Blank UI	ics (v2.0.0)				
Board Name PC Operating System						
Windows Resolution Any						
COLOR DEPTH	WIDTH	HEIGHT				
16 bit		k here to see the				
	CREATE	lable UI Templates				



Documentation Link: Tutorial 1

Step GS-2: Compile and run the example

- 1. Click the "Run Simulator" button to compile and run the example
- 2. Inspect the compiler log by pressing the "Detailed Log" button
- 3. Inspect the project on disk by pressing the "Browse Code" button



Extra

- Try to make small changes in the example, like moving existing widgets or adding new ones.
- Rerun the simulator and see the changes



Step GS-3: Create an application for an STM32 Evaluation Kit (if available)

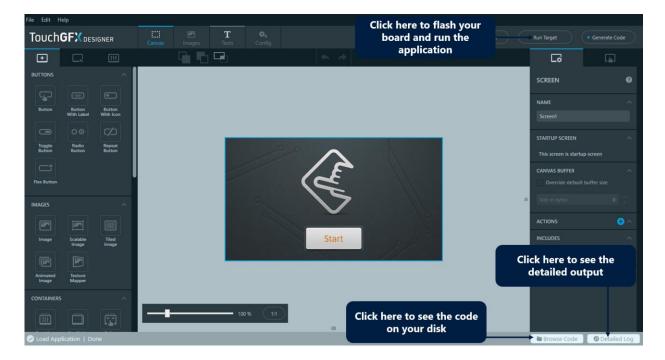
- 1. Create a new project (File/New)
- 2. Click the "Application Template" selection to modify this
- 3. Select the relevant board
- 4. Select an example as the UI Template
- 5. Click the "Create" button

MY APPLICATIONS	ONLINE APPLICATIONS	ONLINE APPLICATIONS					
Create New Application							
APPLICATION NAME	APPLICATION DIRECTORY						
MyAnimatedImageExample746	C:\TouchGFXProjects	a 🖸					
APPLICATION TEMPLATE	UI TEMPLATE	D ment					
Simulator by STMicroelectronics (v2.0.0)	AnimatedImage Example by STMicroelectronics (v1.1.0)	0					
Board Name PC Operating System		D Trees					
Windows Resolution Any CHANG	E Start	D men					
Click here to select	RESOLUTION	O finite					
another Application Template	\$ 480 x 272						
Template	CREATE						



Step GS-4: Compile and flash the STM32 Evaluation Kit

- Connect your STM32 Evaluation Kit to your computer using a USB cable. Make sure to connect it to the ST-Link USB-port.
- 2. Click the "Run Target" button to compile and flash the example (using GCC and STM32 Cube Programmer)
- 3. Inspect the compiler log by pressing the "Detailed Log" button
- 4. Check that the application is running on your board



Troubleshooting

• If you get an error message during flashing try to open Cube Programmer and see if you can connect to the board. Make sure to either disconnect Cube Programmer from the board or close Cube Programmer before retrying to "Run Target" in the Designer.



Documentation Link: Tutorial 1



Step UI-1: Create a blank UI project for your STM32 Evaluation Kit (or simulator if you have no kit available)

- 1. Create a new project (File/New)
- 2. Click the "Application Template" selection to modify this
- 3. Select the relevant board
- 4. Make sure that "Blank UI" is selected as the UI Template
- 5. Click the "Create" button

MY APPLICATIONS	ONLINE APPLICAT	IONS				
Create New Application				Recent Applic	ations	
APPLICATION NAME	APPLICATION DIRE	CTORY		SEARCH FOR RECENT APPLICA	TIONS	
MyApplication1	C:\TouchGFXProj	ects	=			
APPLICATION TEMPLATE	UI TEMPLATE			Materia	Inclusion in the later	1
STM32F746G Discovery Kit by STMicroelectronics (v3.0.0)	Blank UI by STMicroelectron	ics (v2.0.0)	0	Contractory		(
Board Name STM32F746G Operating System FreeRTOS				Control to the Arts		(
Resolution 480 x 272 TouchGFX EMMEATOR				Contract of the line	1	(
COLOR DEPTH	WIDTH	HEIGHT		Contract of the later	1	(
24 bit	480	272		Constant of the local	-	(
	CREATE			Contraction of the local division of the loc	Includes of the local	1

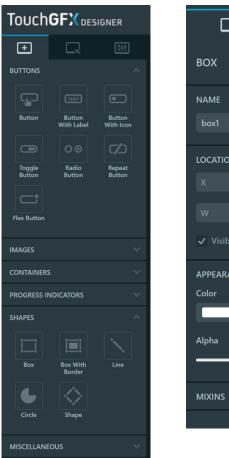


Step UI-2: Insert a Box as background and a Button on top

- 1. Insert a Box widget and expand it to cover the entire canvas (The Box widget can be found in the "Shapes" widget class)
- 2. Change the color of the Box by selecting the Box and changing the "Color" property
- 3. Insert a Button widget and place it in the center of the canvas (The Button widget can be found in the "Buttons" widget class)
- 4. Compile and run on the simulator and run on your board. Check that you can press the button and see it change appearance.

Extra

• Try adding other widgets as well





Step UI-3: Insert an Image

- 1. Acquire a .PNG image either by downloading one or identifying an image locally on you computer
- 2. Add the image to your application (See figure)
- 3. Insert an Image widget (The Image widget can be found in the "Images" widget class)
- 4. Change the inserted Image widget to use the .PNG by changing the image property of the Image widget

Images Image Name Section Extra Section Image Format Size Click the Images tab gpng bg.png ExtFlashSection Default 480 x 272 Click the Images tab Click the plug own_btn_gisabled.png counter_box.png ExtFlashSection Default 152 x 154 Click the plug Click the plug own_btn_gisabled.png e down_btn_gisabled.png ExtFlashSection Default 130 x 56 Click the plug Image format	Edit Help									
Images Image Name Section Extra Section Image Format Size Image Format Size Itab ggng ounter_boxpng own_btn.png own_btn.png bg.png ExtFlashSection ExtFlashSection Default 480 x 272 Click the plu import an import an i				¢. Config				Run Target		
grag Image Name Section Extra Section Image Format Size ounter_boxpng bg.png ExtFashSection ExtFashSection Default 480 x 272 Click the plug image Format Size Size Size Size Click the plug Click the plug image Format Size Size Size Size Size Click the plug image Format Size Size Size Size Size Click the plug image Format Size Size Size Size Size Click the plug image Format Size Size Size Size Size Click the plug image Format Size Size Size Size Size Click the plug import an in Size Size Size Size Size Size import an in Size Size Size Size Size Size import an in Size Size Size Size Size Size import an in Size Size Size Size Size Size import an in Size Size Size Size Size S										Click the Imag
ounter_boxpng bg.png ExtFlashSection ExtFlashSection Default 480 x 272 Click the plu Click the plu lown_btn.png counter_boxpng ExtFlashSection Default 152 x 154 Click the plu Import an in lown_btn.pressed.png down_btn.pressed.png ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in lown_btn.pressed.png down_btn_pressed.png ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in lown_btn_pressed.png down_btn_pressed.png ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in up_btn.png ExtFlashSection ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in up_btn.png ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in up_btn.png ExtFlashSection ExtFlashSection Default 130 x 56 Click the plu Import an in up_btn.png ExtFlashSection Default 130 x 56 Click the plu	🖬 images	Image	Name	Section	Extra Section	Image Format	Size			tab
lown_btn_disabled.png counter_box.png ExtFlashSection ExtFlashSection Default 152 x 154 ExtFlashSection ExtFlashSection Default 130 x 56 import an in down_btn_pressed.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in import disabled.png down_btn_pressed.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in import disabled.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in Import an in import disabled.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in Import an in import disabled.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in import disabled.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in import disabled.png ExtFlashSection ExtFlashSection Default 130 x 56 Import an in	counter_box.png	-	bg.png	ExtFlashSection	n ExtFlashSection	Default	480 x 272			
image: state of the state	down_btn.png down_btn_disabled.png		counter_box.png	ExtFlashSection	n ExtFlashSection		152 x 154			Click the plus
p_bth_disabled.png	down_btn_pressed.png up_btn.png		down_btn.png		n ExtFlashSection		130 x 56			import an ima
up_btn.png ExtFlashSection Default 130 x 56	up_btn_disabled.png		<u>down_btn_disabled.png</u>	ExtFlashSection	ExtFlashSection		130 x 56			
up btn disabled.ong ExtFlashSection ExtFlashSection Default 130 x 56		-						-		Imported ima
panel		-	up_btn_disabled.png	ExtFlashSection	n ExtFlashSection		130 x 56			
								Browse Code	Detailed Log	

Extra

• Try adding other images as well



Step UI-4: Inspect the generated code

- 1. Click the "Browse Code" button
- Open the code files in the MyApplication/TouchGFX/GUI/ folder. Either by opening the project files for Visual Studio or CubeIDE or by opening the relevant .cpp and .hpp files in an editor.
 Relevant files:
 - TouchGFX/gui/include/gui/screen1_screen/Screen1View.hpp
 - TouchGFX/gui/src/screen1_screen/Screen1View.cpp
 - TouchGFX/generated/gui_generated/include/gui_generated/screen1_screen/Screen1ViewBase.hpp
 - TouchGFX/generated/gui_generated/src/screen1_screen/Screen1ViewBase.cpp
- 3. The ViewBase class is created, owned and updated by the Designer
- 4. The View class is owned and updated by the developer
- 5. View class inherits from the ViewBase class
- 6. Add a new widget to the application. Click "Generate Code" or "Run Simulator" and inspect the Screen1ViewBase.hpp file to see that it now has a new member. Representing the new widget.

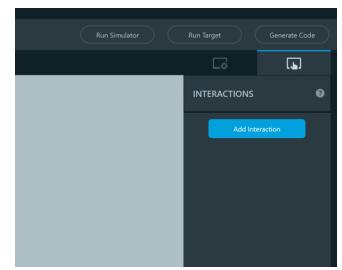


Step UI-5: React to Button click by adding an Interaction

- 1. Select the Interaction tab and press "Add Interaction"
- 2. Select:
 - Trigger = "Button Clicked"
 - Choose clicked source = "button1" (or whatever you have named your button)
 - Action = "Move Widget"
 - Choose widget to move = "image1" (or whatever you have named your image)
 - X = 10, Y = 10, Duration = 2000ms
- 3. Run the application and see that the image is moving when you press the button

Extra

• Try other settings for the interaction e.g. "Easing" which will change the movement behavior

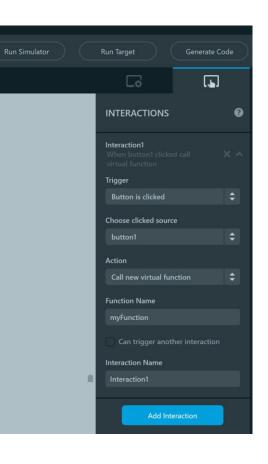




Step UI-6 (1/2): React to Button click in user code

- 1. Select the Interaction tab and press "Add Interaction"
- 2. Select:
 - Trigger = "Button Clicked"
 - Choose clicked source = "button1" (or whatever you have named your button)
 - Action = "Call new virtual function"
 - Function name = "myFunction"
 - Open source file TouchGFX/generated/gui_generated/include/ gui_generated/screen1_screen/Screen1ViewBase.hpp
 - Observe the new function myFunction()







Step UI-6 (2/2): React to Button click in user code

- 1. Open source file TouchGFX/gui/include/gui/screen1_screen/Screen1View.hpp
- 2. Override the myFunction() from the Screen1ViewBase class in the Screen1View class
- 3. Implement changing color of the background in myFunction()
 - Note that the implementation here is done in the header file for the sake of simplicity. Normally you would do this in the source file.

```
class Screen1View : public Screen1ViewBase
{
  public:
    Screen1View();
    virtual ~Screen1View() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
    virtual void myFunction()
    {
        box1.setColor(box1.getColor() + 1234);
        box1.invalidate();
    }
```

Extra

- Do other things in myFunction() e.g. change alpha value for the image or button (hint: use .setAlpha(uint8_t)) (not the background box since alpha values for background widget will result in strange behavior)
- Try the printf debug functionality by inserting: touchgfx_printf("New color value: %i\n", box1.getColor());
 in myFunction().



Documentation Link: Tutorial 2

Thank you

© STMicroelectronics - All rights reserved. The STMicroelectronics corporate logo is a registered trademark of the STMicroelectronics group of companies. All other names are the property of their respective owners.

