TouchGFX Workshop

UI development – Getting Started

life.augmented

Agenda

1 Introduction

2 Installation

3 Getting started

4 Creating your own UI application

Introduction

Introduction

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

Introduction

Further reading

- You will find a lot of help afterwards in the TouchGFX documentation site: https://support.touchgfx.com/
- 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: Getting started - what´s next

Introduction

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

Documentation Link: Development Introduction

Installation

Installation

- 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)

Documentation Link: Installation

Installation

- 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.

Documentation Link: Installation

Installation

- 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

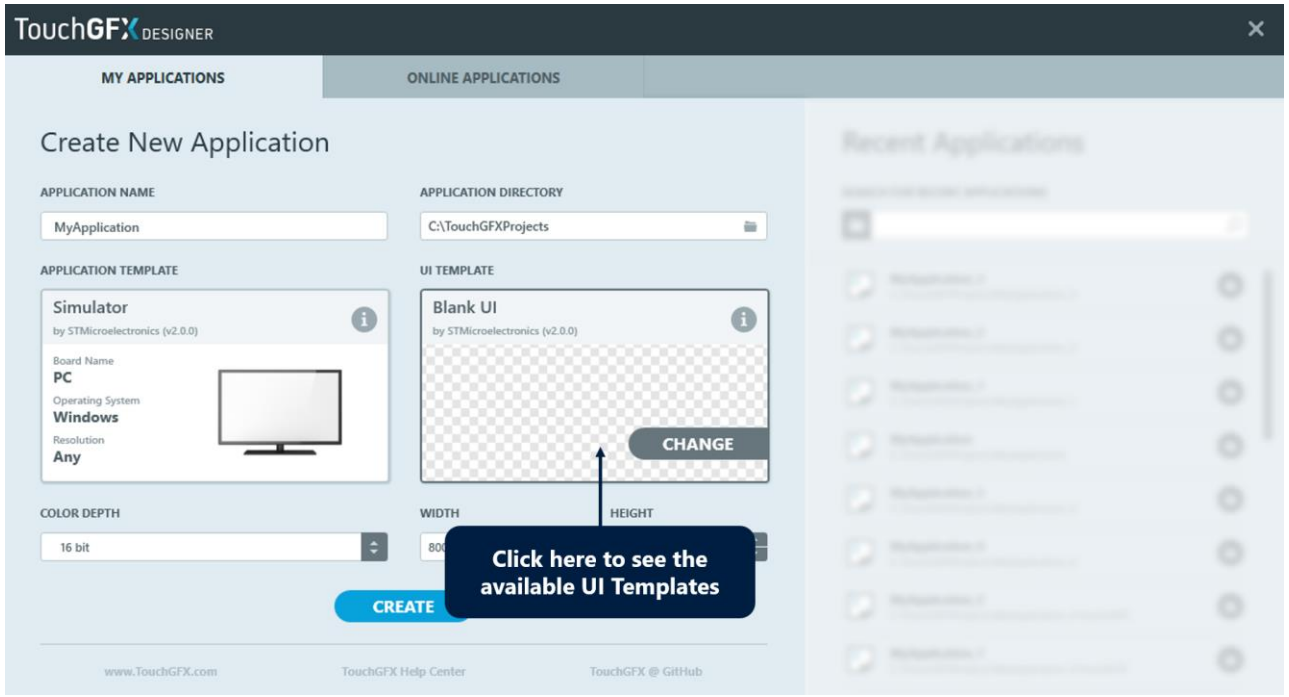
Documentation Link: Installation

Getting started

Getting started

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

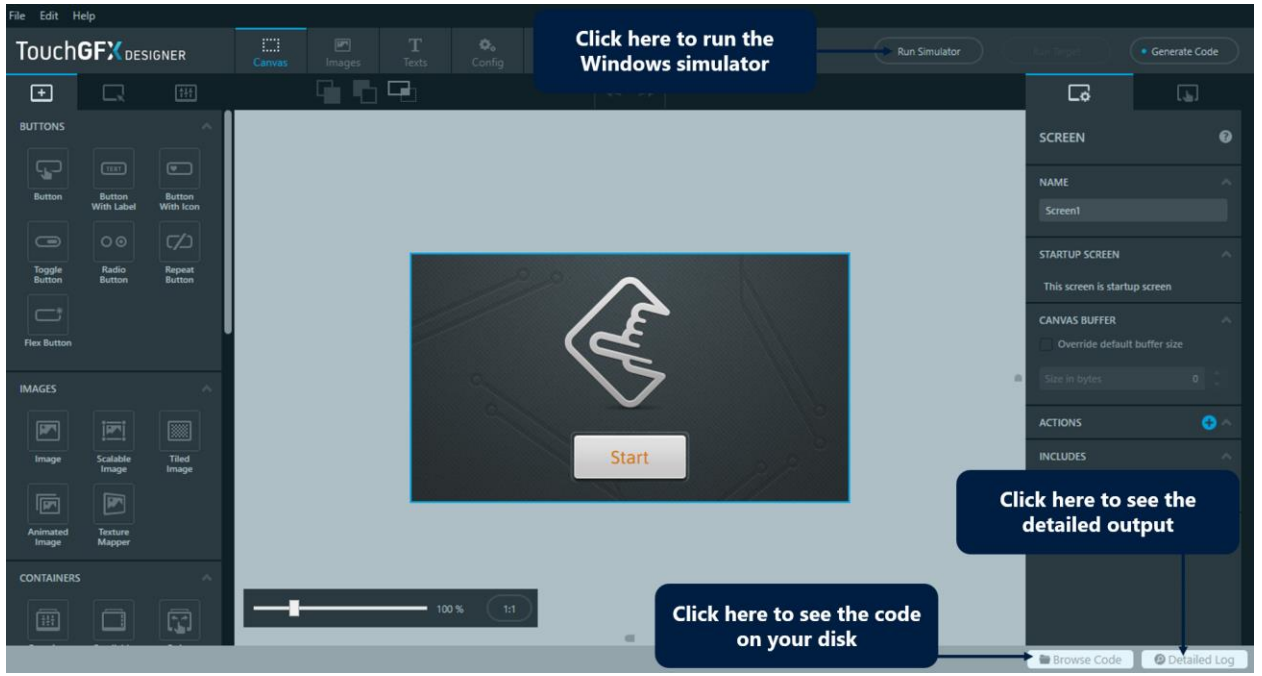


Documentation Link: Tutorial 1

Getting started

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

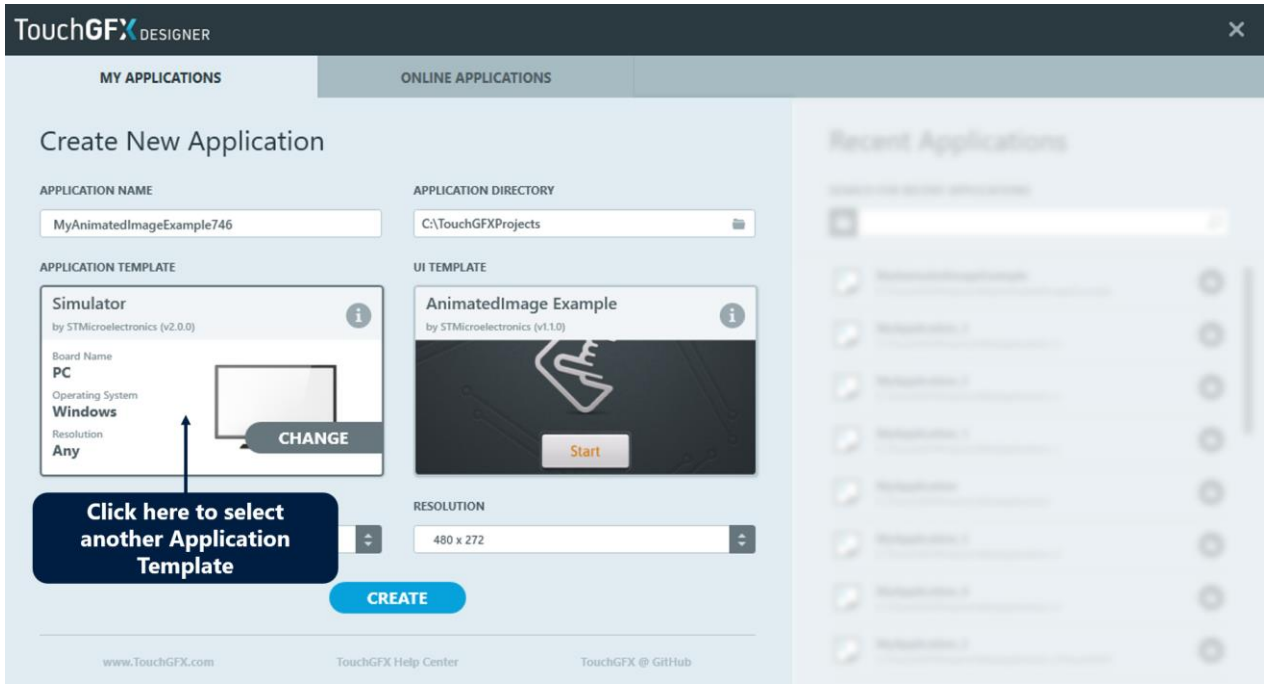
- *Try to make small changes in the example, like moving existing widgets or adding new ones.*
- *Rerun the simulator and see the changes*

Documentation Link: Tutorial 1

Getting started

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

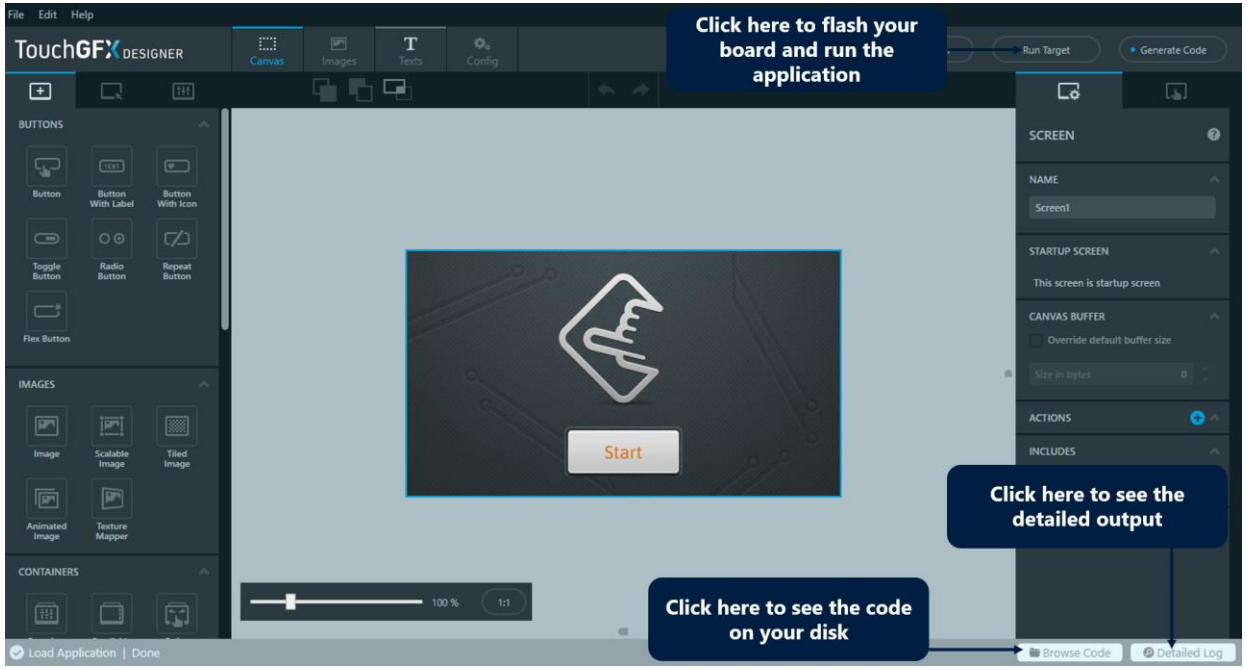
1. Create a new project (File/New)
2. Click the “Application Template” selection to modify this
3. Select the relevant board
4. Select an example as the UI Template
5. Click the “Create” button



Documentation Link: Tutorial 1

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

1. 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.*

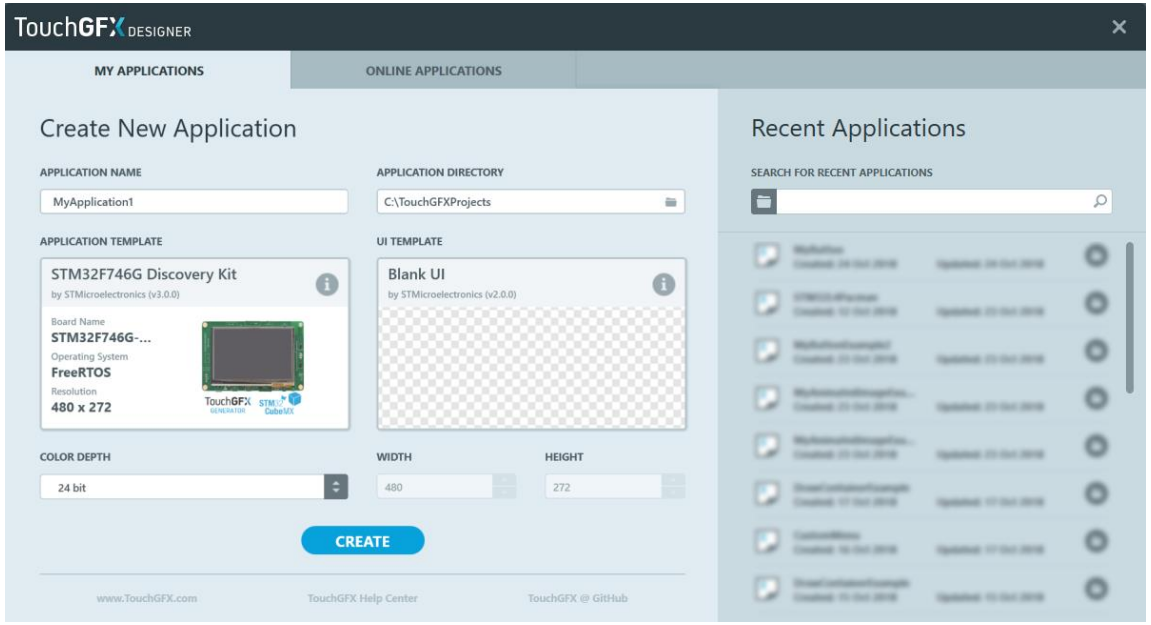
Documentation Link: Tutorial 1

Creating your own UI application

Creating your own UI application

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



Documentation Link: Tutorial 2

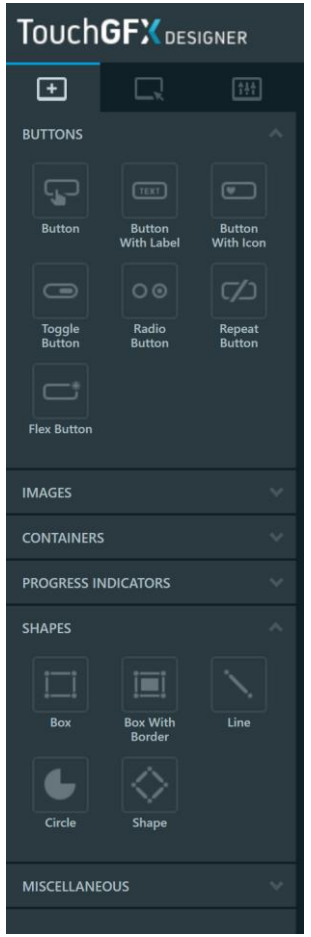
Creating your own UI application

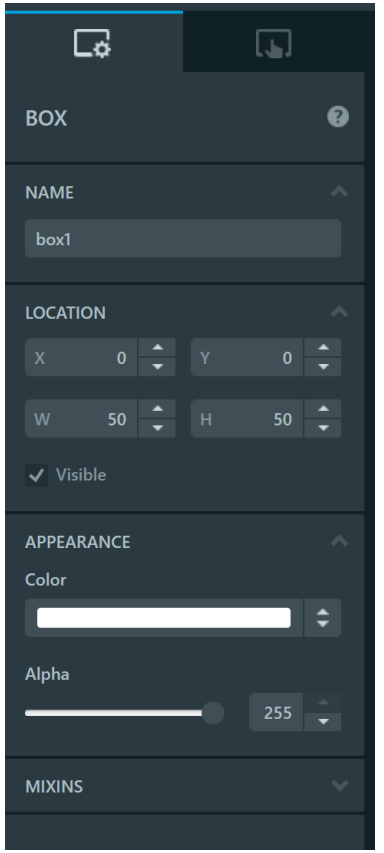
Step UI-2: Insert a Box as background and a Button on top

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*



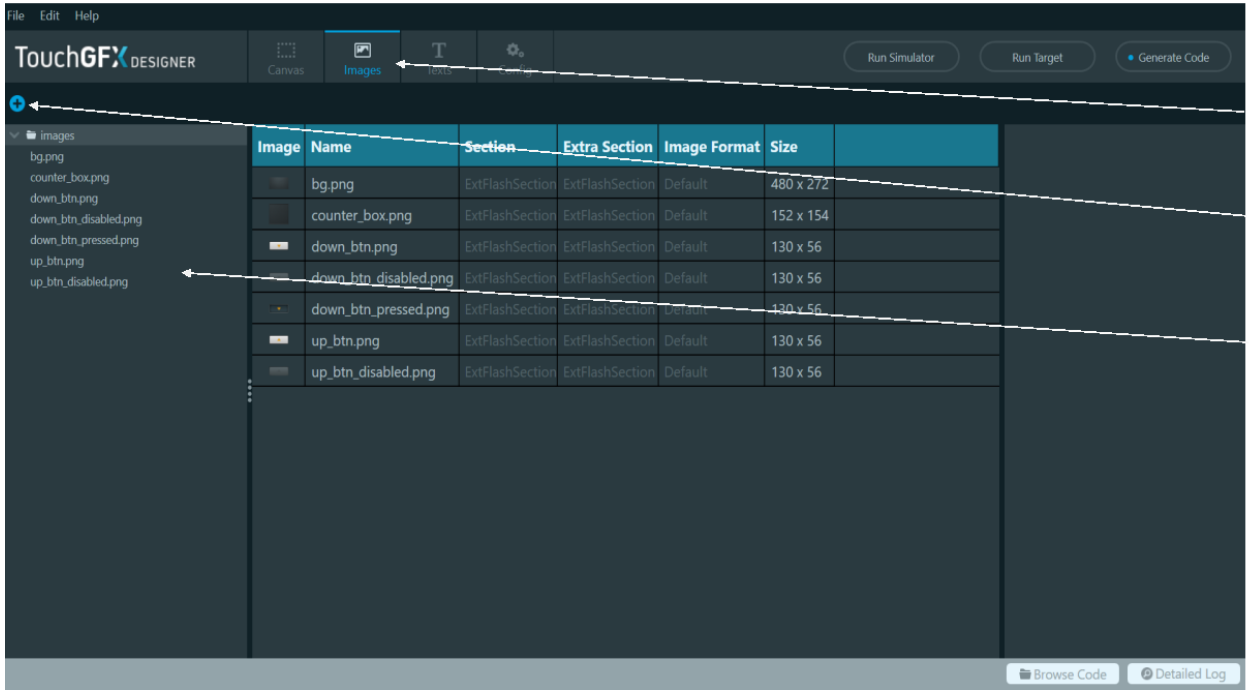


Documentation Link: Tutorial 2

Creating your own UI application

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



Extra

- *Try adding other images as well*

Documentation Link: Tutorial 2

Creating your own UI application

Step UI-4: Inspect the generated code

1. Click the “Browse Code” button
2. 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:

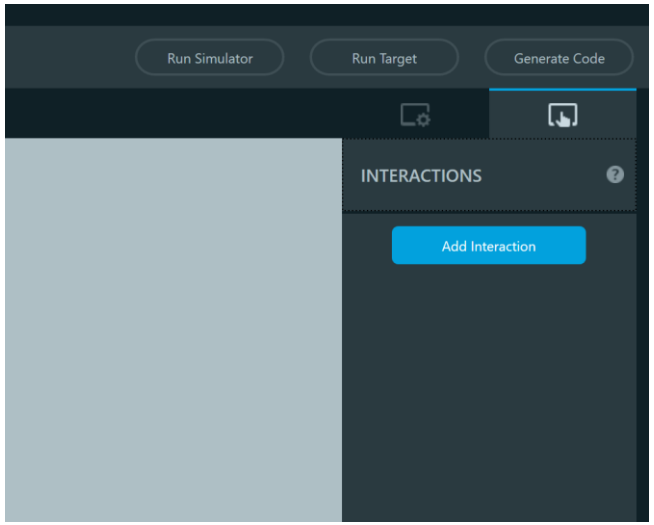
   - `TouchGFX/gui/include/gui/screen1_screen/Screen1View.hpp`
   - `TouchGFX/gui/src/screen1_screen/Screen1View.cpp`
   - `TouchGFX/generated/gui_generated/include/gui_generated/screen1_screen/Screen1ViewBase.hpp`
   - `TouchGFX/generated/gui_generated/src/screen1_screen/Screen1ViewBase.cpp`

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.

Documentation Link: Tutorial 2

Creating your own UI application

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

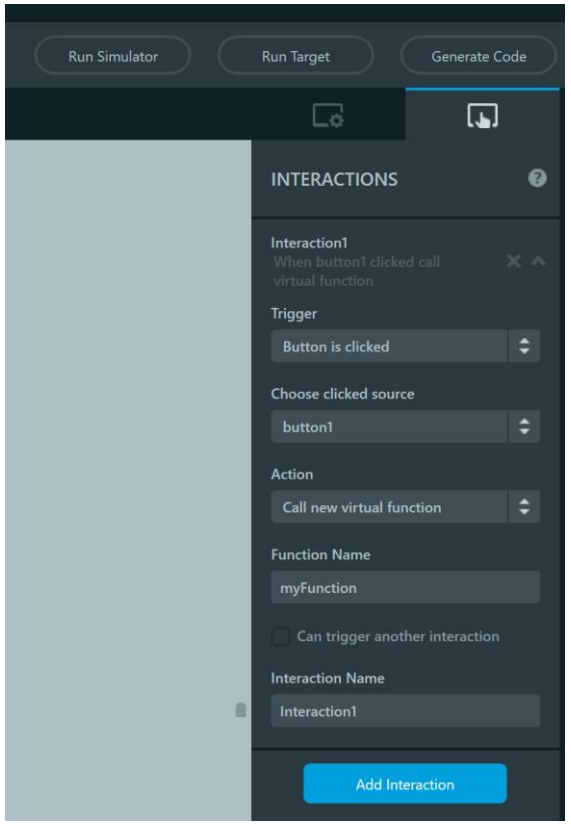
- *Try other settings for the interaction e.g. “Easing” which will change the movement behavior*

Documentation Link: Tutorial 2

Creating your own UI application

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()




```
class Screen1ViewBase : public touchgfx::View<Screen1Presenter>
{
public:
    Screen1ViewBase();
    virtual ~Screen1ViewBase() {}
    virtual void setupScreen();

    /*
     * Virtual Action Handlers
     */
    virtual void myFunction()
    {
        // Override and implement this function in Screen1
    }
```

Documentation Link: Tutorial 2

Creating your own UI application

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

life.augmented

Thank you

life.augmented