life.augmented

Welcome to **EURO23 workshop**

Graphics track

Workshop team

Introduction

- The purpose of this hands-on session is to give you a demonstration about how to start with the evaluation and developments with STM32U5x9 and NeoChrom GPU
- We will use a STM32U5A9J-DK evaluation board and the PC software TouchGFX Designer
- Unbox the Discovery kit, connect USB cable to PC, and we are ready to start!

Legenda

- Slides including following symbol are purely theoretical ones
- Optional steps during demo setup are marked with a grey bar
- Yellow bar shows tips & tricks, reminders, shortcuts, ideas, etc

something

- Source code for development is included inside pink boxes

```
HAL_Delay(500);
```

Agenda

- 1 TouchGFX Designer intro
  5mins
- 2 Setup the screen
  15mins
- 3 Setup the interactions
  20mins
- 4 Compile and Run
  5mins
- 5 Move to C++ world
  10mins
- 6 Takeaways, Links, Q&A
  5mins

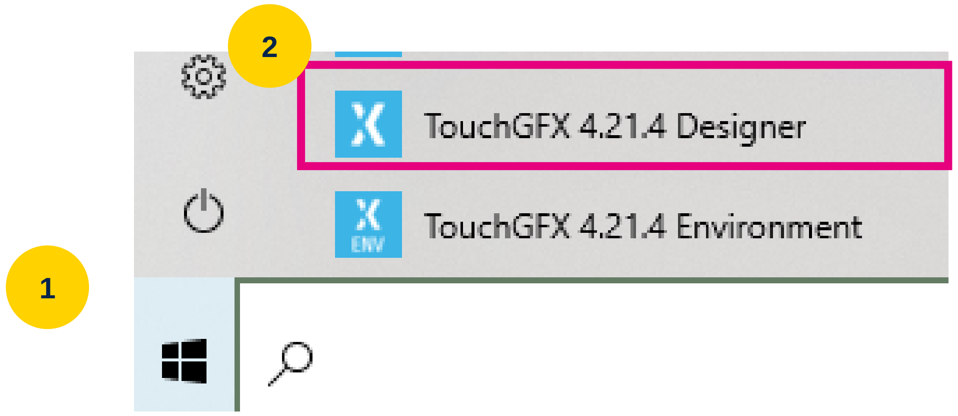
life.augmented

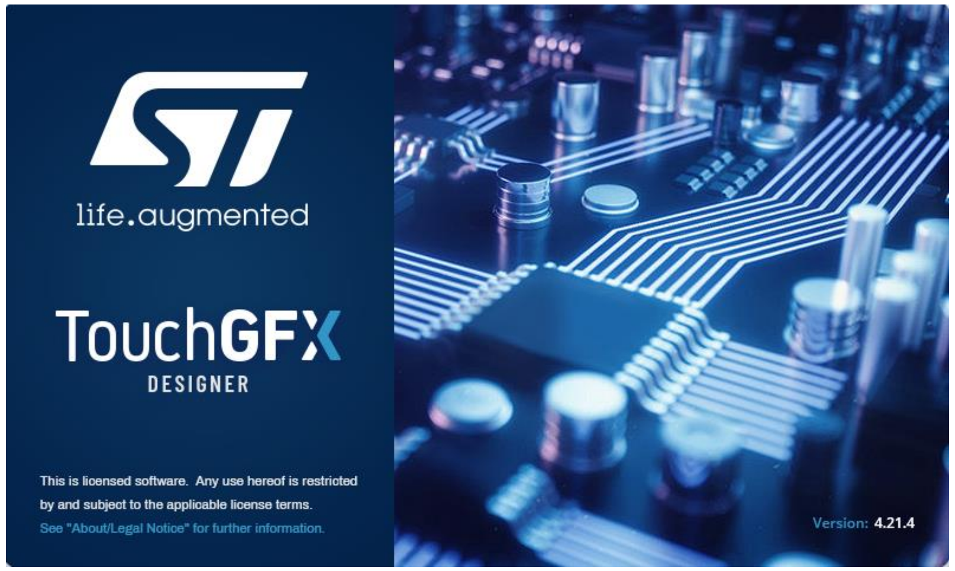
TouchGFX Designer intro

life.augmented

Open the TouchGFX Designer tool

1. Click on **Start**, scroll to 'S', expand "**STMicroelectronics**"
2. Launch the **TouchGFX 4.21.4 Designer** PC GUI

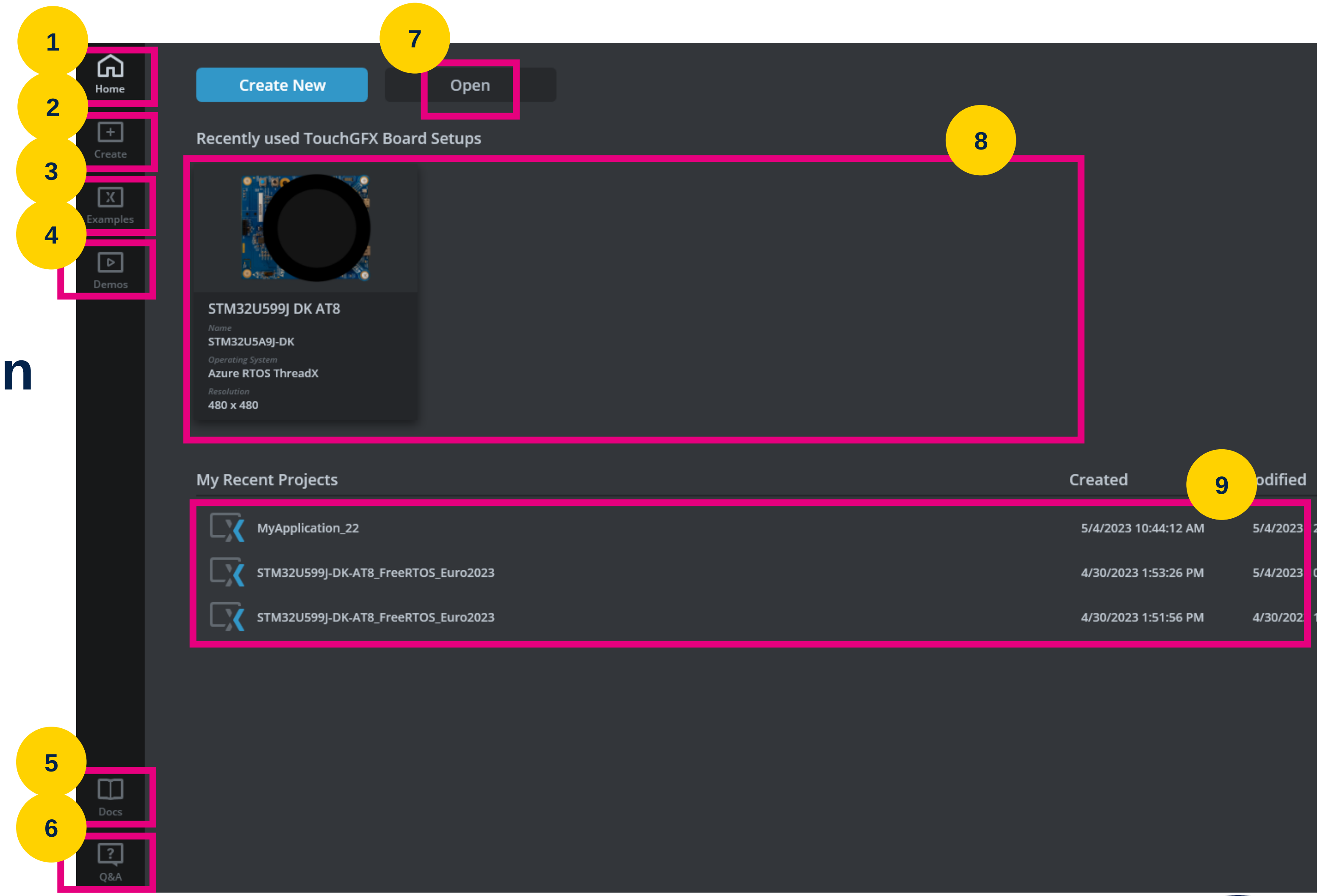




This is licensed software. Any use hereof is restricted by and subject to the applicable license terms.
See "About/Legal Notice" for further information.

TouchGFX Designer: how it looks like #1

1. This screen
2. Start with a new project
3. Select a **widget** example
4. Select a complete **application**
5. Go to online Support
6. Go to online Community
7. Open an old project
8. History of used **TBSs**
9. History of opened projects



STM32U599J DK AT8
Name
STM32U5A9J-DK
Operating System
Azure RTOS ThreadX
Resolution
480 x 480

	Created	odified
MyApplication_22	5/4/2023 10:44:12 AM	5/4/2023
STM32U599J-DK-AT8_FreeRTOS_Euro2023	4/30/2023 1:53:26 PM	5/4/2023
STM32U599J-DK-AT8_FreeRTOS_Euro2023	4/30/2023 1:51:56 PM	4/30/202

Widgets & TBSs will be explained into following slides

TouchGFX building blocks #1: the TBS

- A TouchGFX Board Setup (TBS) includes all the **Board Initialization Code** needed to prototype on a standard STM32 Evaluation Kits available out of the box
- The TBSs are provided with a STM32CubeMX project, so it is possible for you to modify the configuration if you want to experiment or add access to more peripherals
- There are currently listed 4 TBSs for STM32U5:
  - STM32U575ZI-NUCLEO (no LTDC/NeoChrom inside) + X-Nucleo-GFX02Z1
    - 2 different controller on Expansion Board options
    - Using STM32CubeMX 6.8 + TouchGFX 4.21.4 + STM32U5 CubeFW 1.1.1
  - STM32U5x9J-DK (LTDC+DSI/NeoChrom inside)
    - 2 Operating Systems supported
    - Using STM32CubeMX 6.8 + TouchGFX 4.21.4 + STM32U5 CubeFW 1.2.0

We'll use TBS for STM32U599J-DK$AT8 with Azure RTOS ThreadX OS

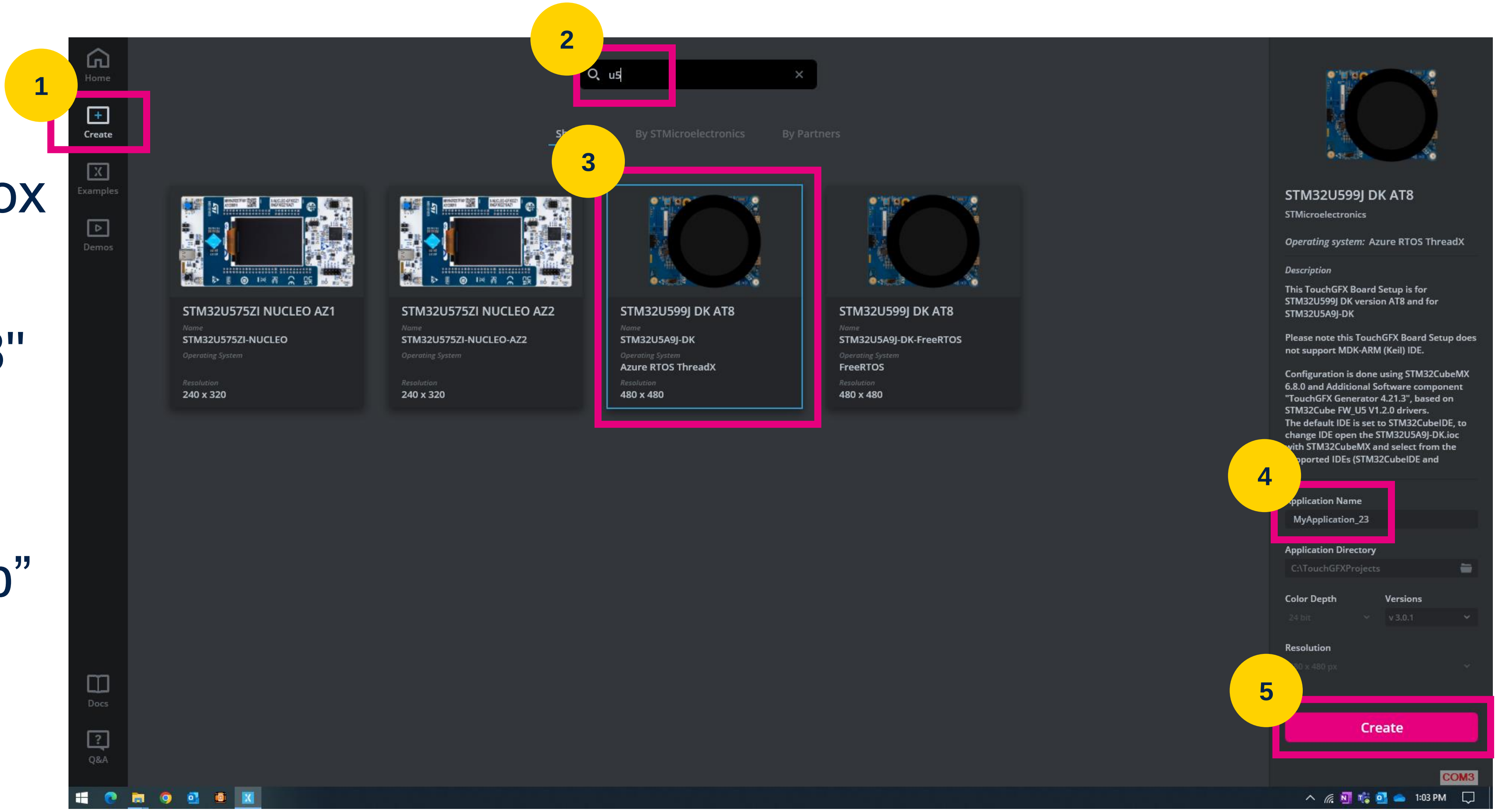
TouchGFX building blocks #2: the Widget

- A Widget in TouchGFX is something that can be drawn on the screen and can be interacted with
    - Buttons, Images, Progress Indicators, Shapes (box, line, circle, etc), Containers
- Using TouchGFX Designer, users can add available widgets to their screens and customize them how they want with the supplied properties specific to each widget
- The order in which you add the widgets will determine the z-order.
- In the widgets list of a given screen, the first widget will cover the other widgets

We'll use Widgets for Buttons, Image, Text, Shape and Texture Mapper

Select the right TBS

1. Click on **Create**
2. Type “u5” on search box
3. Select the third TBS: "STM32U5x9J DK AT8" with ThreadX
4. Type “NeoChrom_Workshop” as name
5. Click on **Create**



STM32U575ZI NUCLEO AZ1
Name: STM32U575ZI-NUCLEO
Operating System:
Resolution: 240 x 320

STM32U575ZI NUCLEO AZ2
Name: STM32U575ZI-NUCLEO-AZ2
Operating System:
Resolution: 240 x 320

STM32U599J DK AT8
Name: STM32U5A9J-DK
Operating System: Azure RTOS ThreadX
Resolution: 480 x 480

STM32U599J DK AT8
Name: STM32U5A9J-DK-FreeRTOS
Operating System: FreeRTOS
Resolution: 480 x 480

STM32U599J DK AT8
STMicroelectronics
Operating system: Azure RTOS ThreadX

Description

This TouchGFX Board Setup is for STM32U599J DK version AT8 and for STM32U5A9J-DK

Please note this TouchGFX Board Setup does not support MDK-ARM (Keil) IDE.

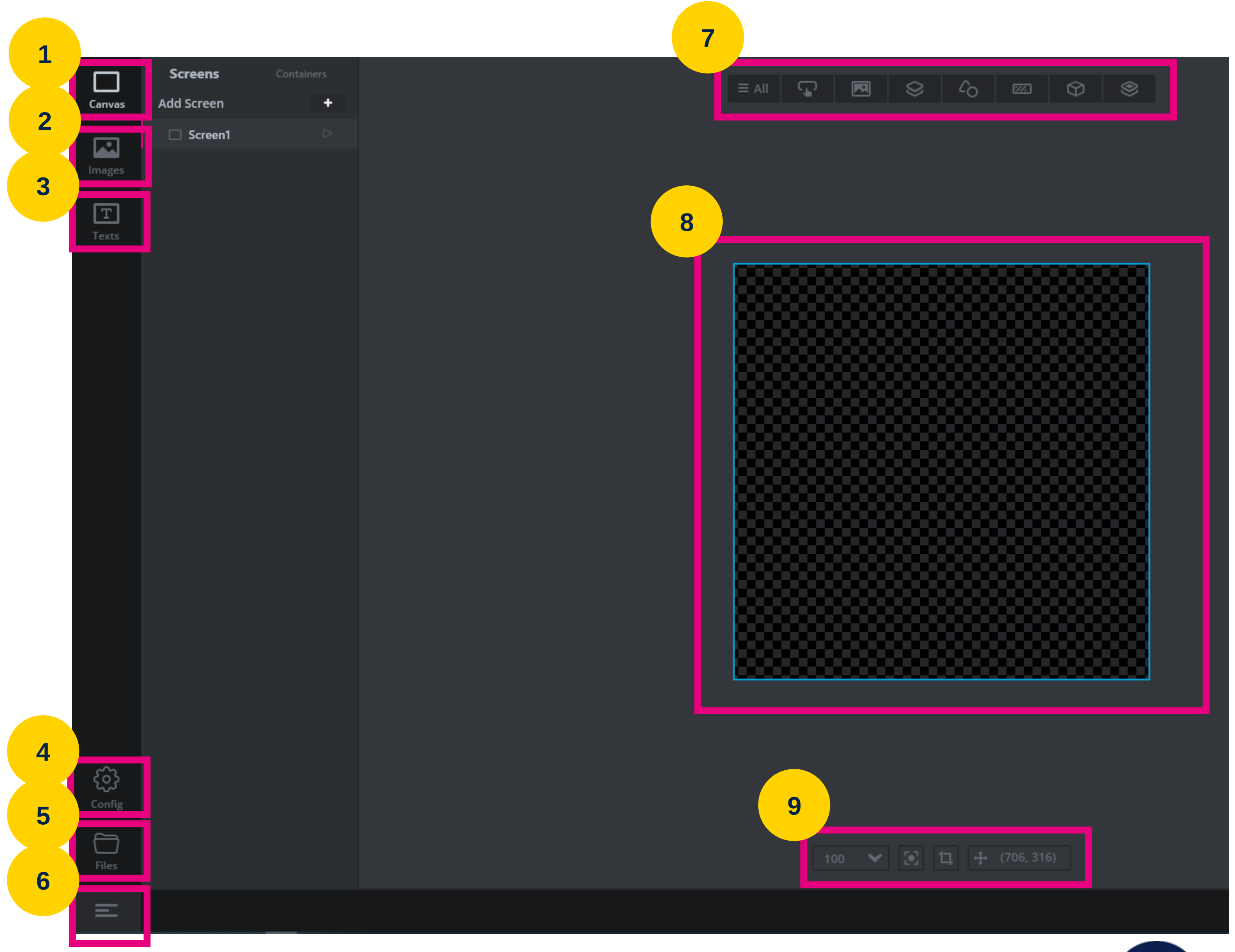
Configuration is done using STM32CubeMX 6.8.0 and Additional Software component "TouchGFX Generator 4.21.3", based on STM32Cube FW_U5 V1.2.0 drivers.
The default IDE is set to STM32CubeIDE, to change IDE open the STM32U5A9J-DK.ioc with STM32CubeMX and select from the pported IDEs (STM32CubeIDE and

pplication Name: MyApplication_23
Application Directory: C:\TouchGFXProjects
Color Depth: 24 bit
Versions: v 3.0.1
Resolution: 0 x 480 px

Create

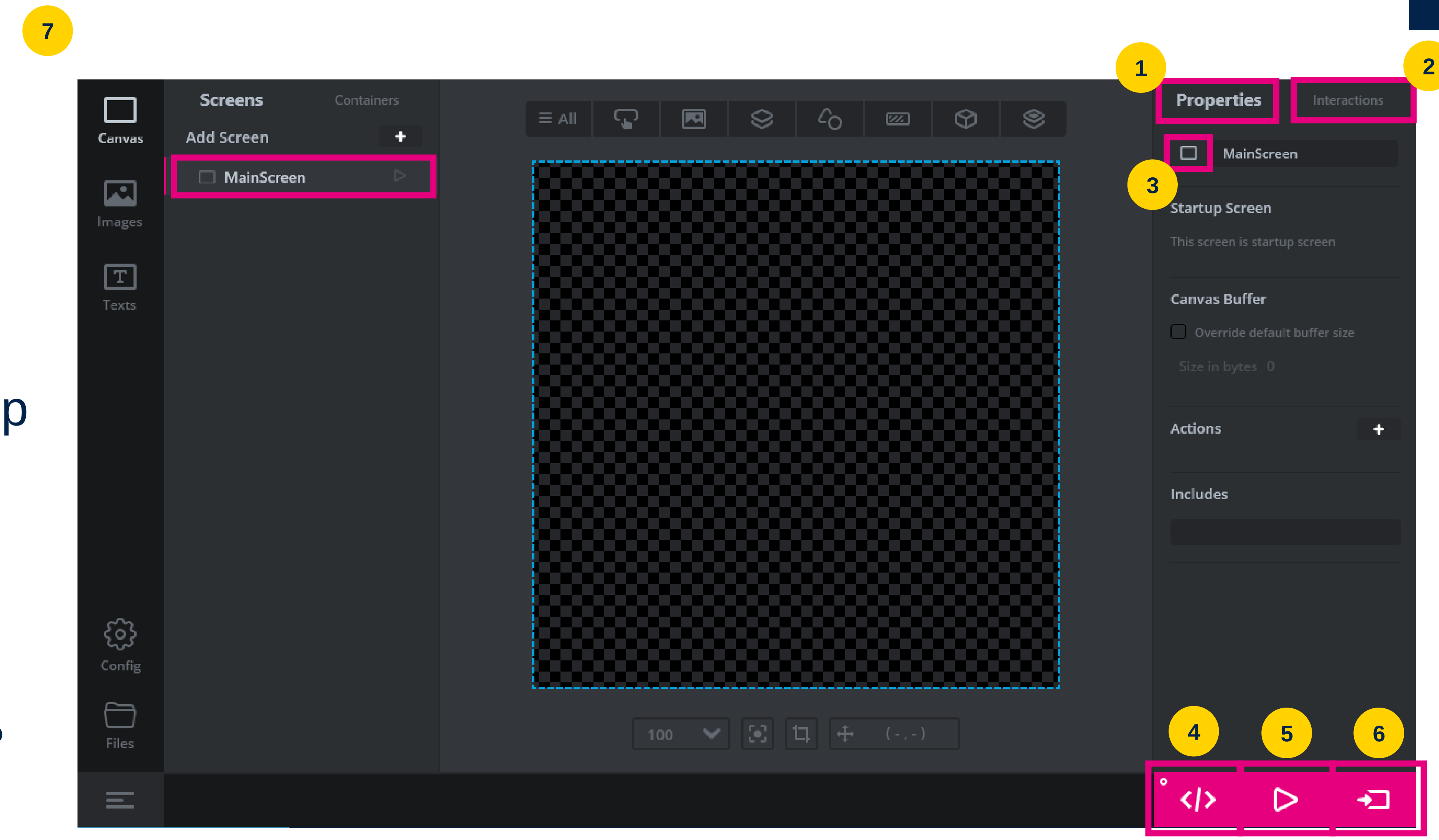
TouchGFX Designer: how it looks like #2

1. Screen list
2. Image assets
3. Textual assets
4. Project configuration
5. Project file system
6. Show compile and Flash output
7. Widgets (incl. search box)
8. Content of the selected screen
9. Helping tools



TouchGFX Designer: how it looks like #3

1. Selected element properties tab
2. Current screen Interactions tab
3. Link to Widget help
4. Code generation
5. Launch Simulator
6. Flash the EvalKit
7. Can you see this?



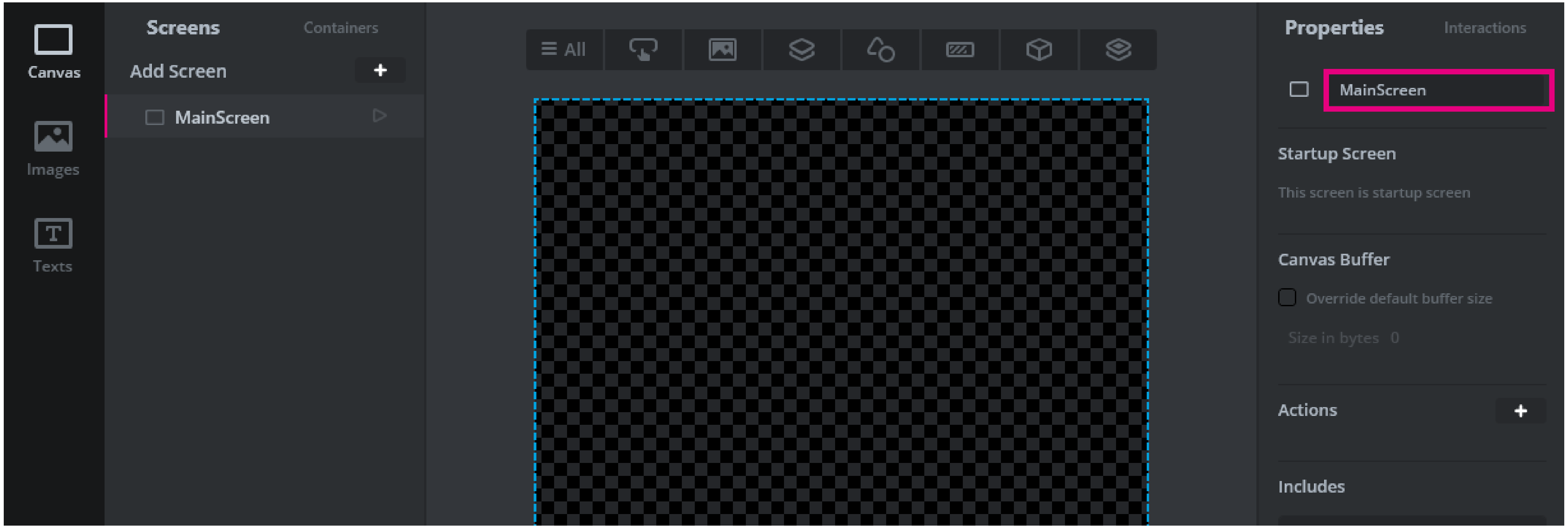
Try the Properties tab: change default name “Screen1” to “MainScreen” (7)

Setup the screen

life.augmented

Let's start with the UI development

Rename screen as "MainScreen"



MainScreen

Startup Screen

This screen is startup screen

Canvas Buffer

Override default buffer size

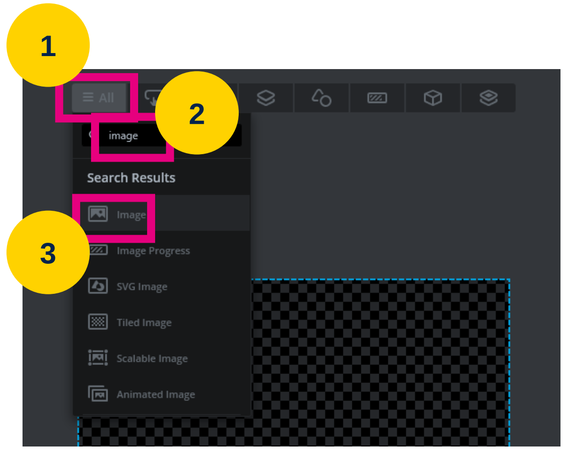
Size in bytes 0

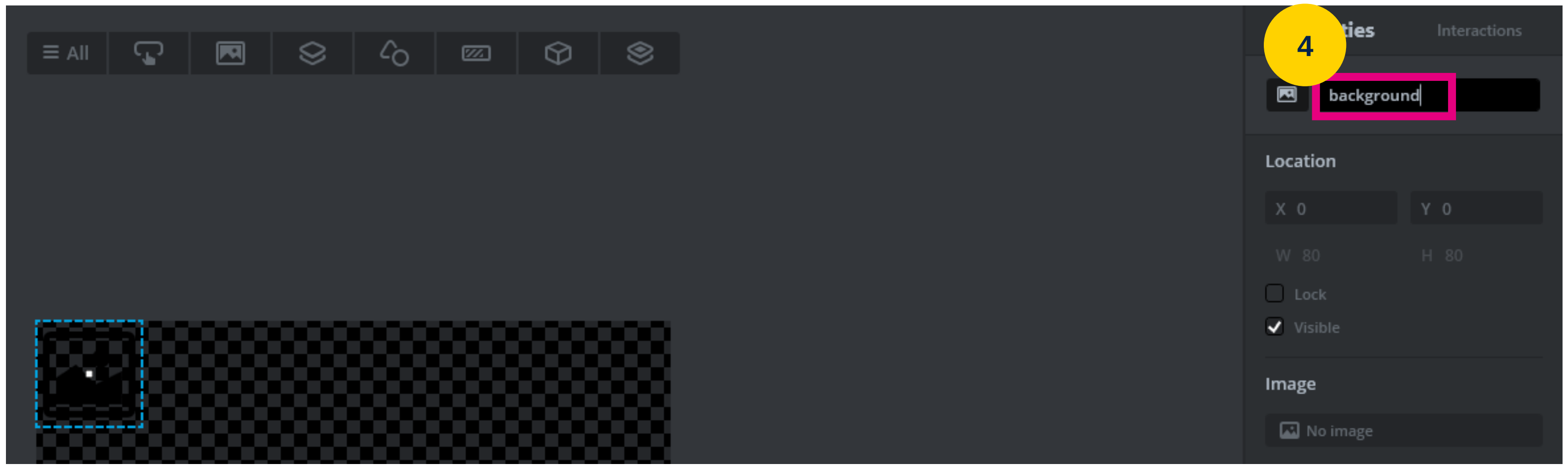
Actions

Includes

Let’s start with the UI development

1. Mouse over “All” Widget Menu
2. Type “image”
3. Select **Image** on result
4. Change default name “Image1” to “background” into Properties tab

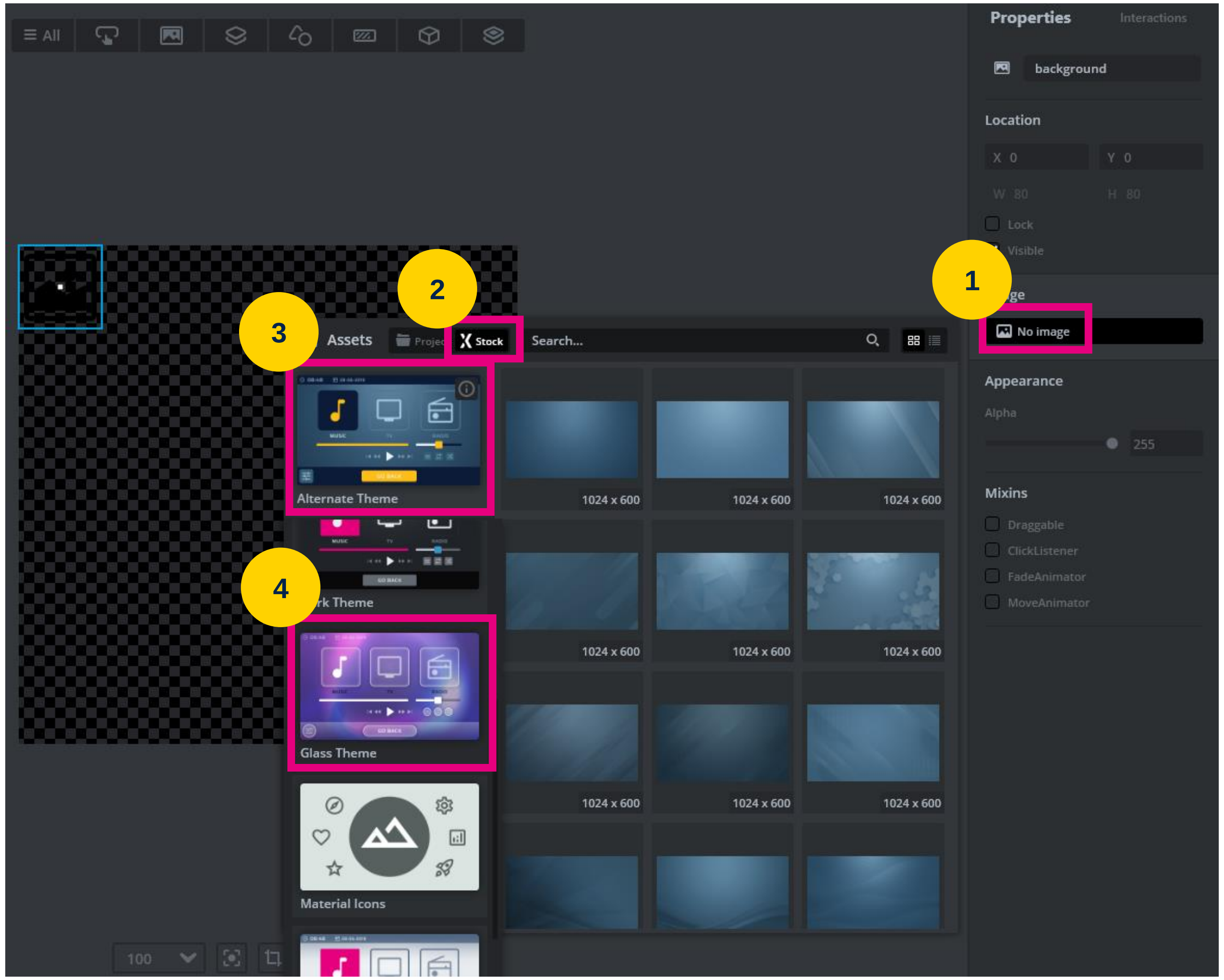




You can directly go to ad-hoc widget menu (vs “All”) if you know in which category it is

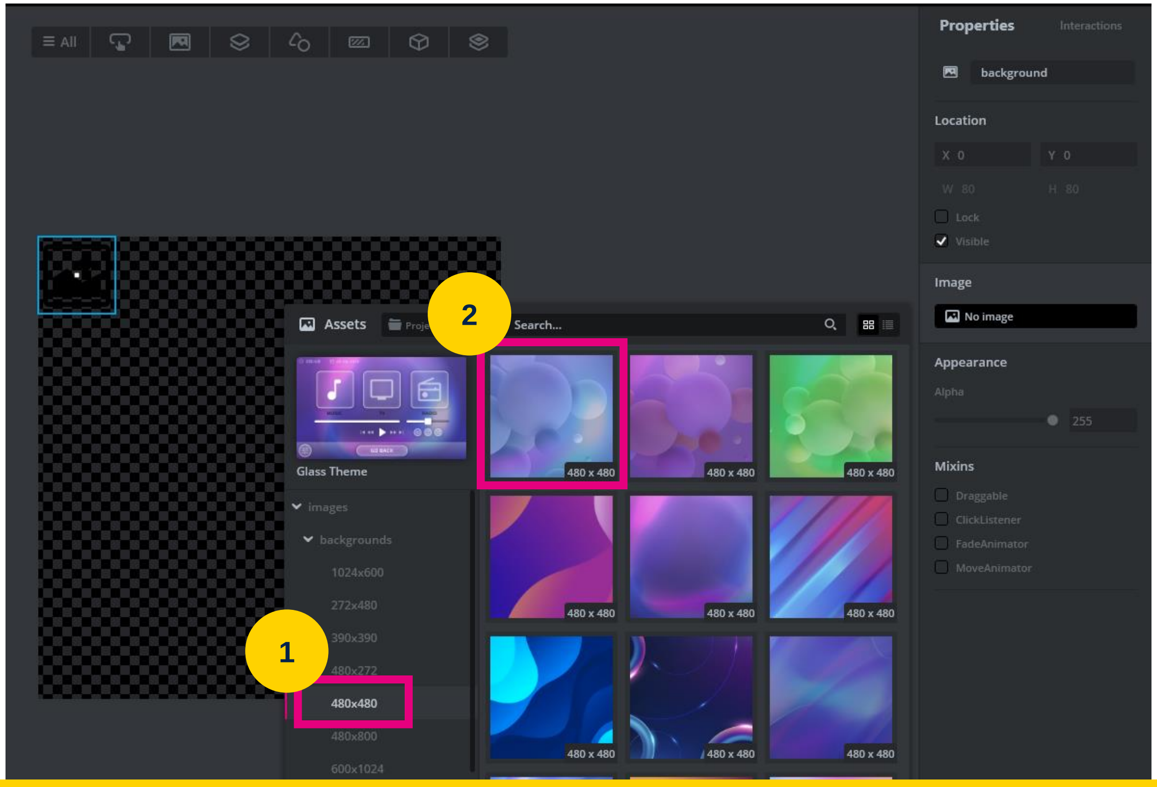
Setup the background

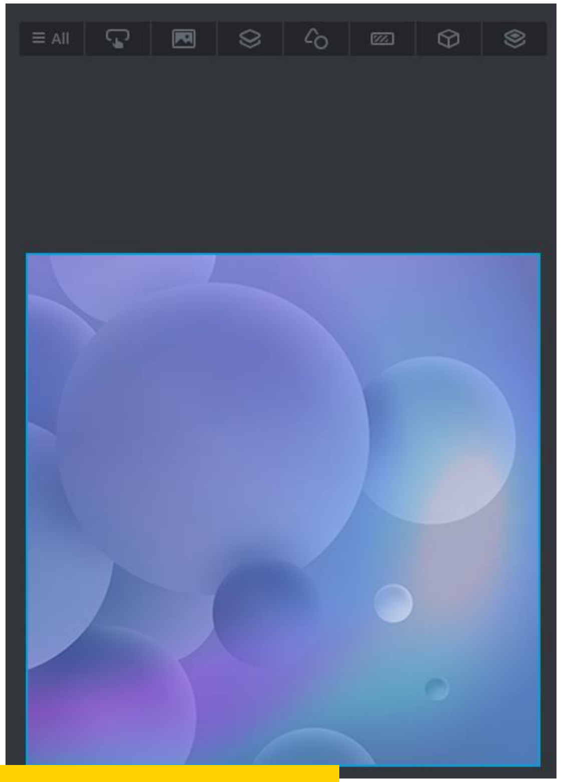
1. Click on "**No image**" in Image section of Properties tab
2. Click on **Stock**
3. Click on default "**Alternate Theme**"
4. Scroll the list with mouse wheel till "**Glass Theme**", and select this new skin



Select background image from Stock

1. Select **480x480** section
2. Select “bubbles_blue.png”

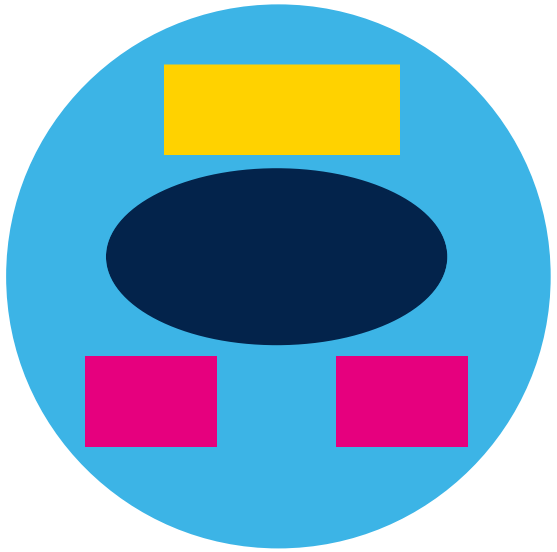




Check and congrats: you added your first Widget on TouchGFX Designer!

Think about the result at first

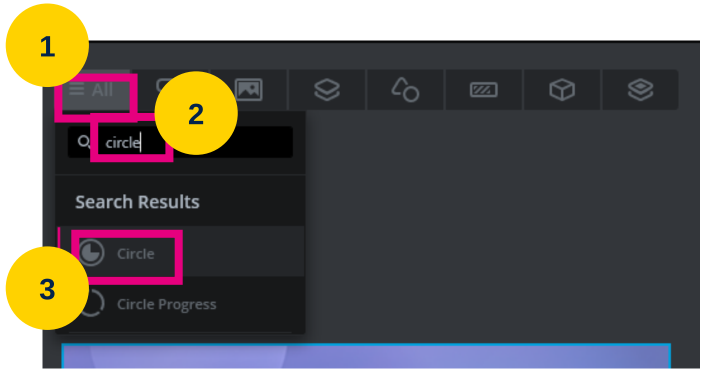
- Other than background, we want to add:
  1. A “temporary” circle to emulate a round display
  2. A text area
  3. Couple buttons with icon from Stock images
  4. A texture mapper
- Remember z-order (Cf. slide #9): first in list is in foreground
  - Widget on top will be rendered front-most on the screen

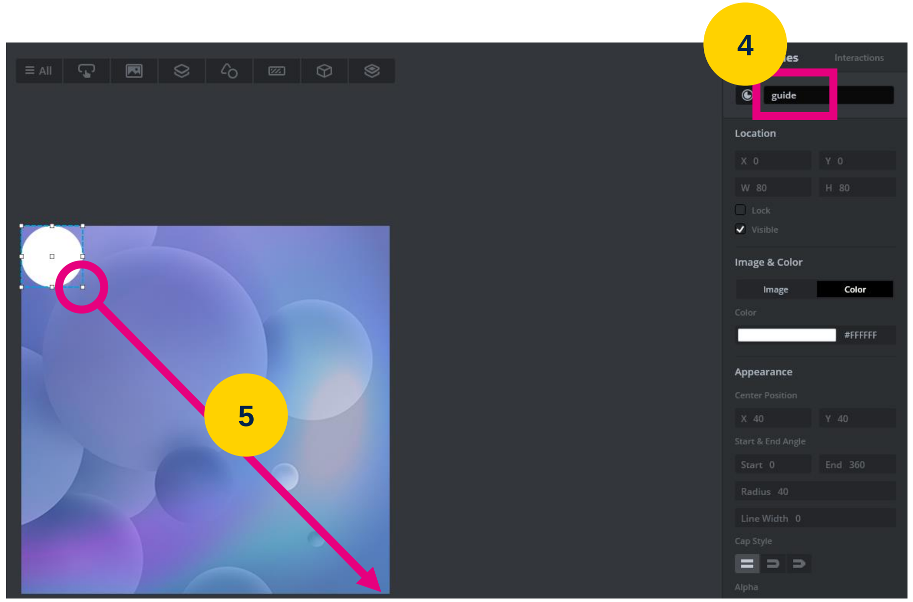


There will be a colored flag on slide’s side to indicate in which of 4 steps we are

Setup screen guide (1/2)

1. Mouse over “All” Widget Menu
2. Type “circle”
3. Select **Circle** on result
4. Change default name “Circle1” to “guide” into Properties tab
5. Drag&Drop bottom-right corner of the circle’s square to maximize its size

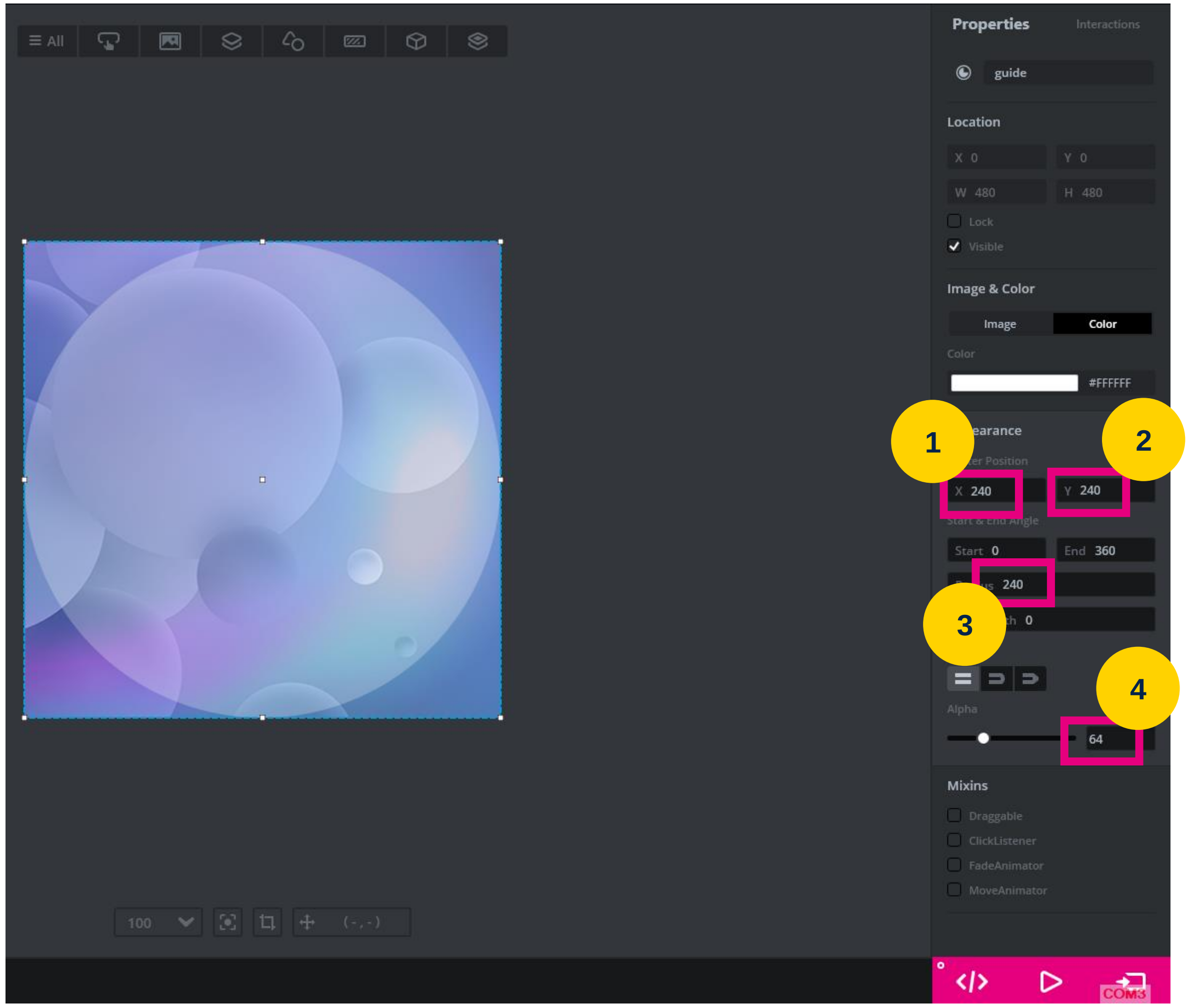




Setup screen guide (2/2)

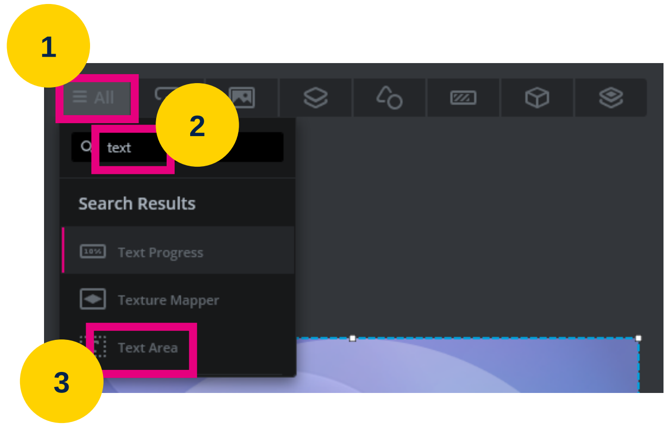
1

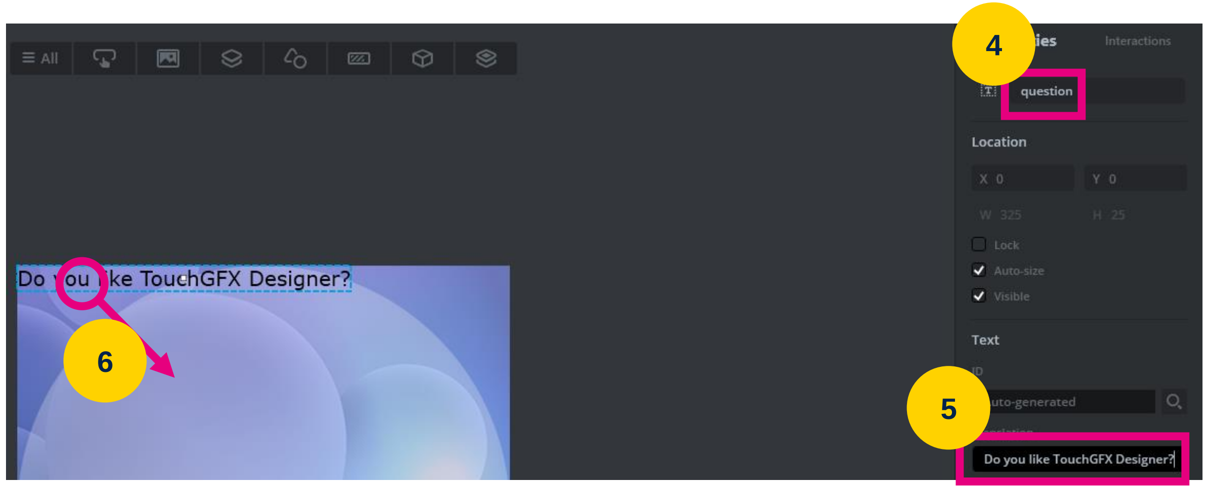
1. Set **Center Position X** to 240
2. Set **Center Position Y** to 240
3. Set **Radius** to 240
4. Set **Alpha** to very small value (e.g. 64) to make the circle transparent



Display some text

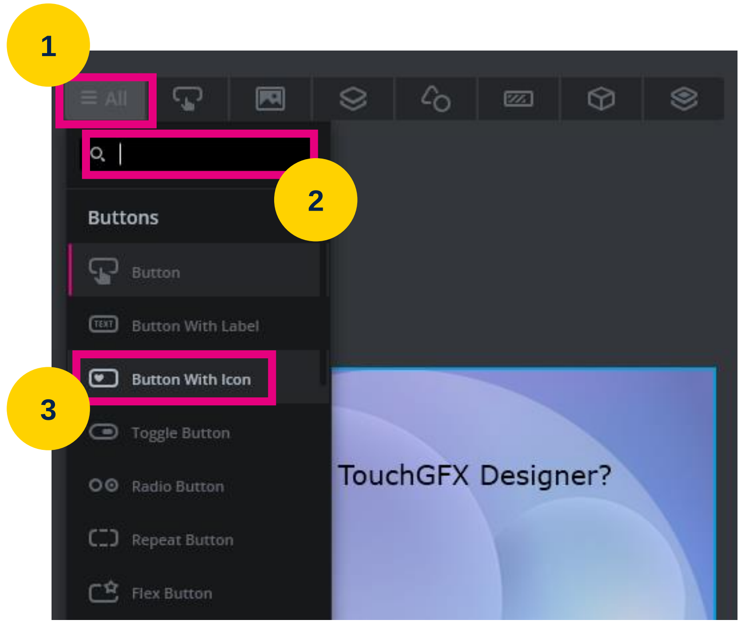
1. Mouse over “All” Widget Menu
2. Type “text”
3. Select **Text Area** on result
4. Change default name “textArea1” to “question” into Properties tab
5. Set **Translation** to any text (e.g. “Do you like TouchGFX Designer?”)
6. Drag&Drop text area to be inside the “guide” circle

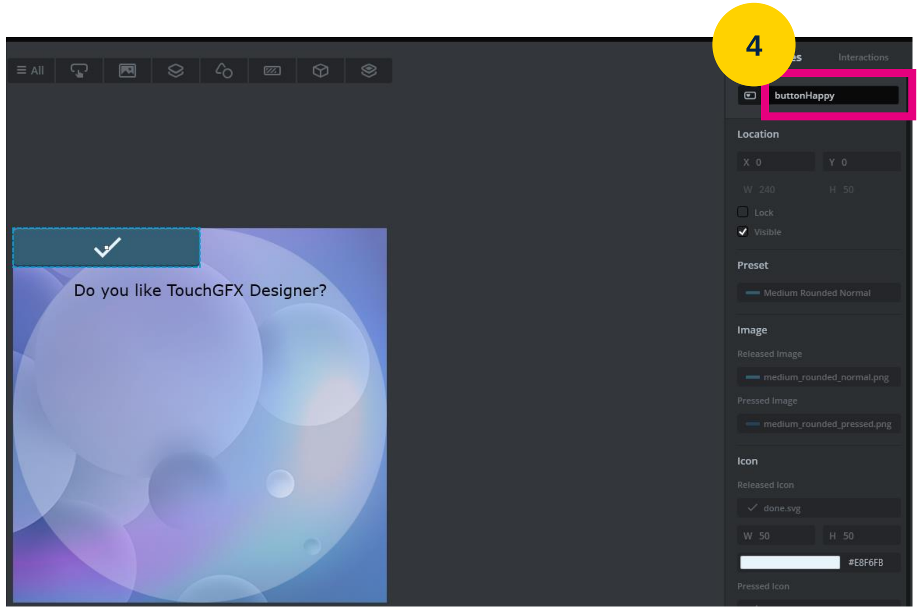




Setup button #1 (1/3)

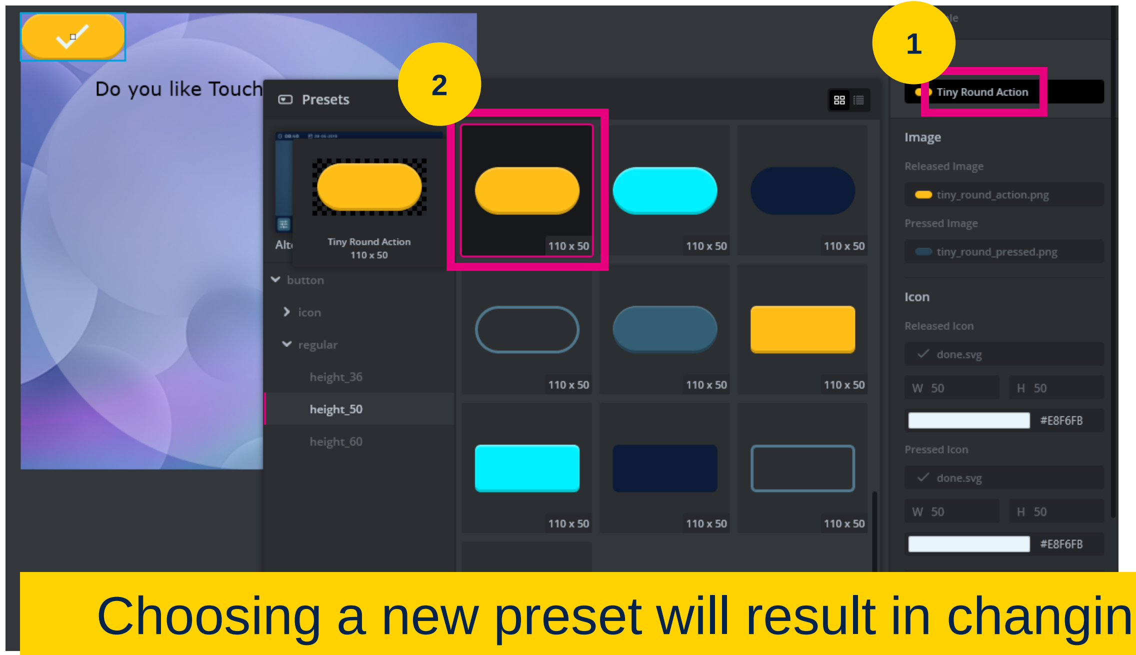
1. Mouse over “All” Widget Menu
2. Type “button”
3. Select **Button With Icon** on result
4. Change default name “buttonWithIcon1” to “buttonHappy” into Properties tab

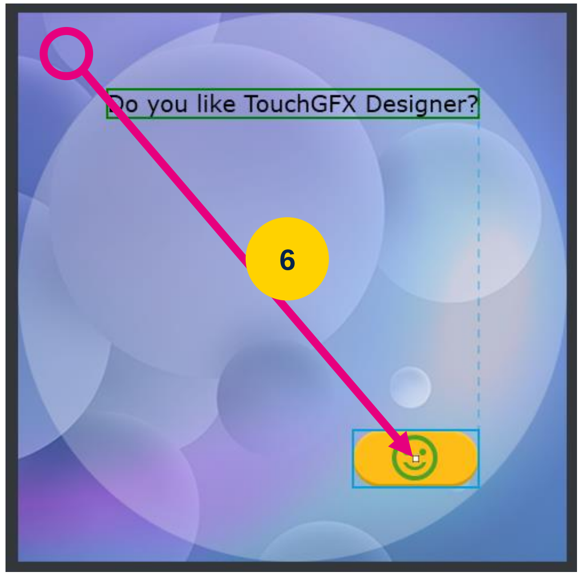




Setup button #1 (2/3)

1. Click on **Preset** into Properties tab
2. Scroll the list with mouse wheel till yellow "Tiny Round Action" 110x50, and select this new preset for both Pressed and Released images
3. Drag&Drop buttonHappy to be inside the "guide" circle

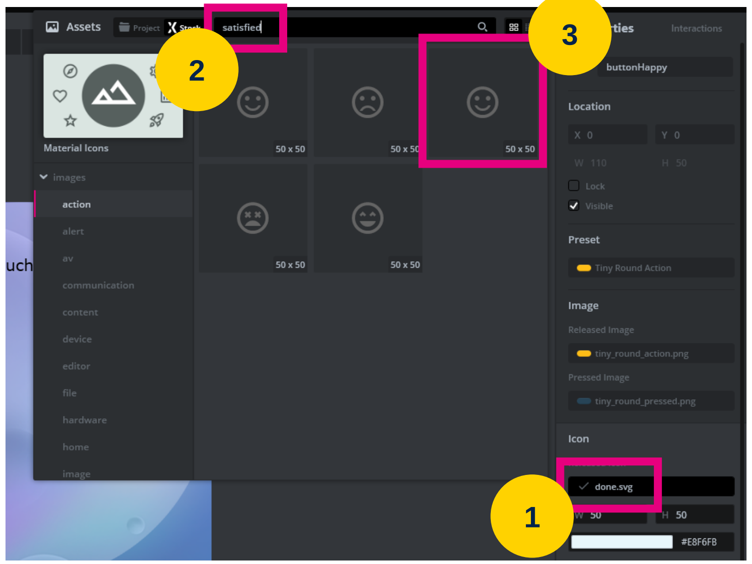


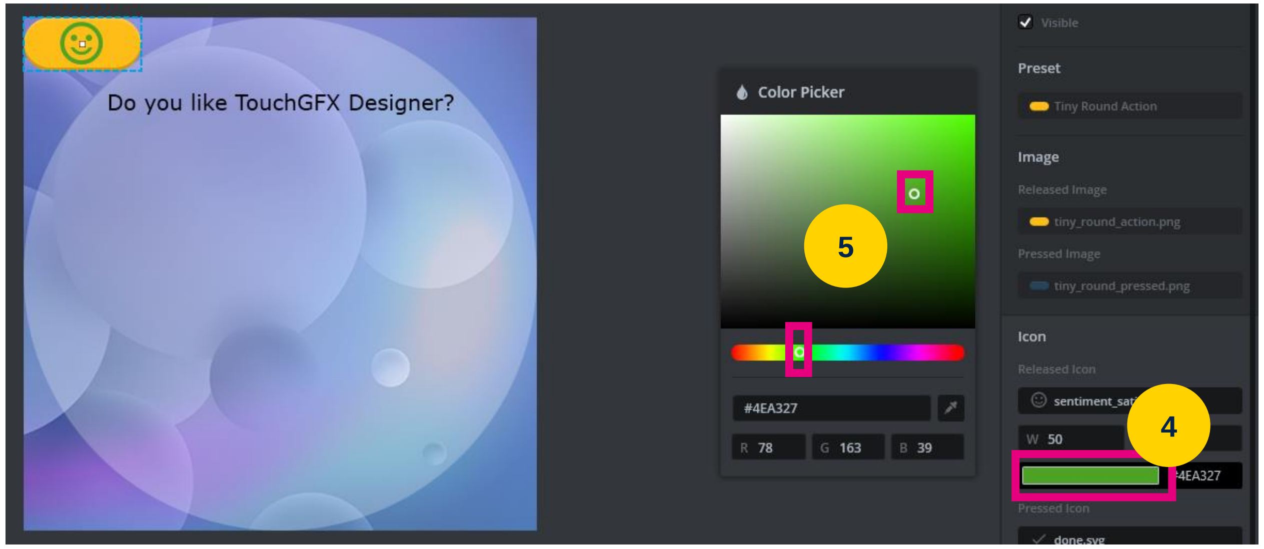


Choosing a new preset will result in changing both Pressed and Released images

Setup button #1 (3/3)

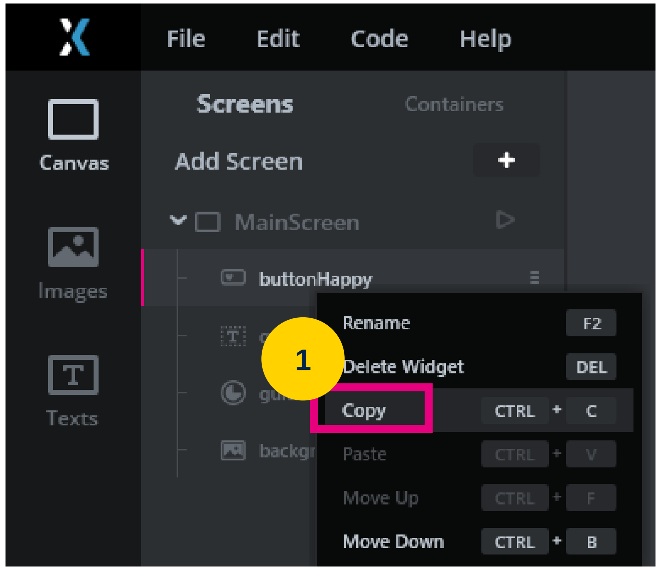
1. Click on **Released Icon** in Icon section of Properties tab
2. Type “satisfied”
3. Select sentiment_satisfied.svg (the smile) on result
4. Click on *Color Picker* in Icon section of Properties tab
5. Select **Green** as color by mouse click

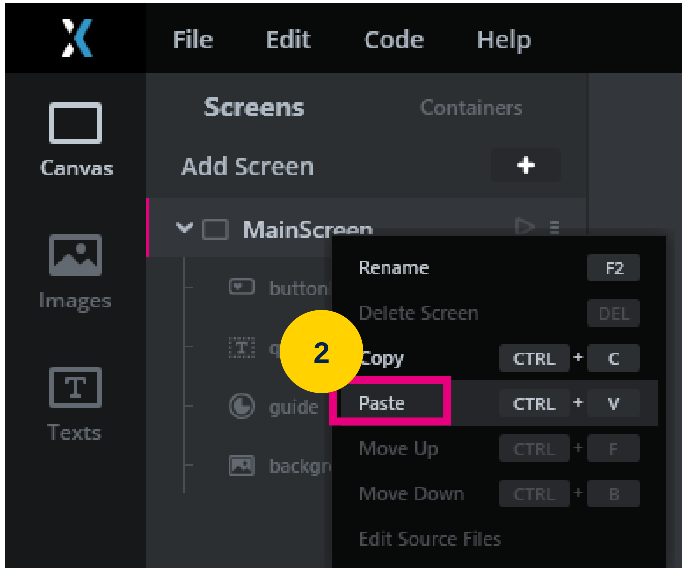


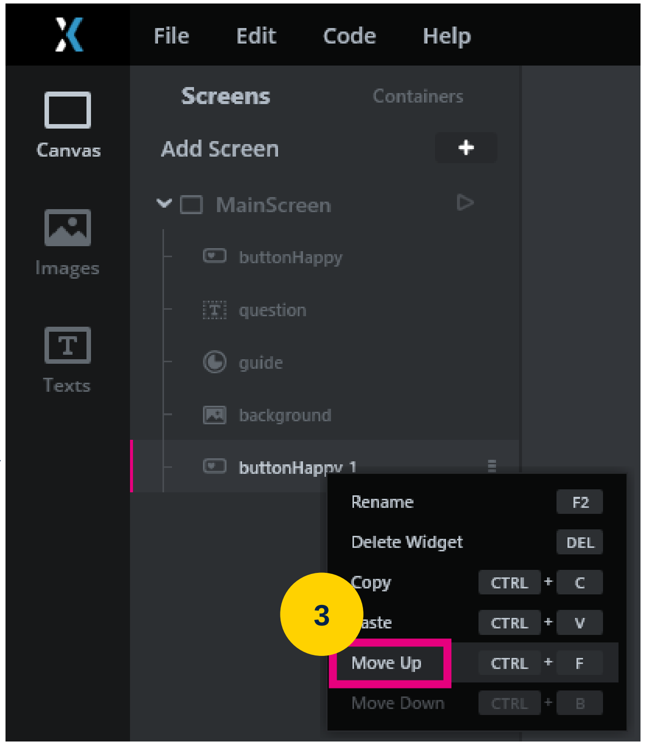


Setup button #2 (1/2)

1. Right-click on **buttonHappy** component, and select Copy
2. Right-click on **MainScreen** component, and select Paste
3. Right-click on **buttonHappy_1**, and select Move Up
   - Repeat this step till the top

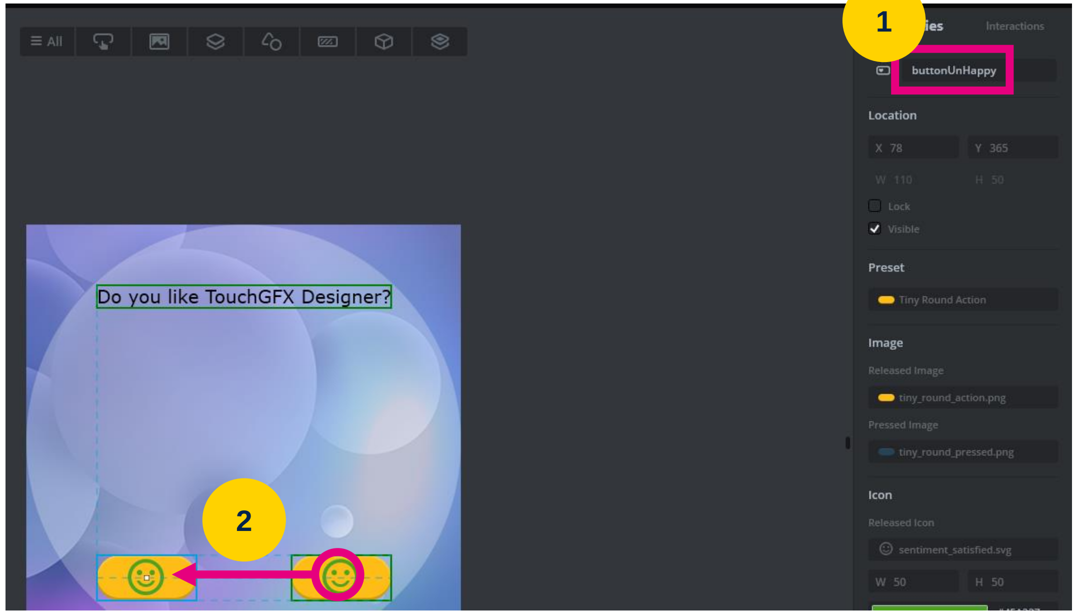




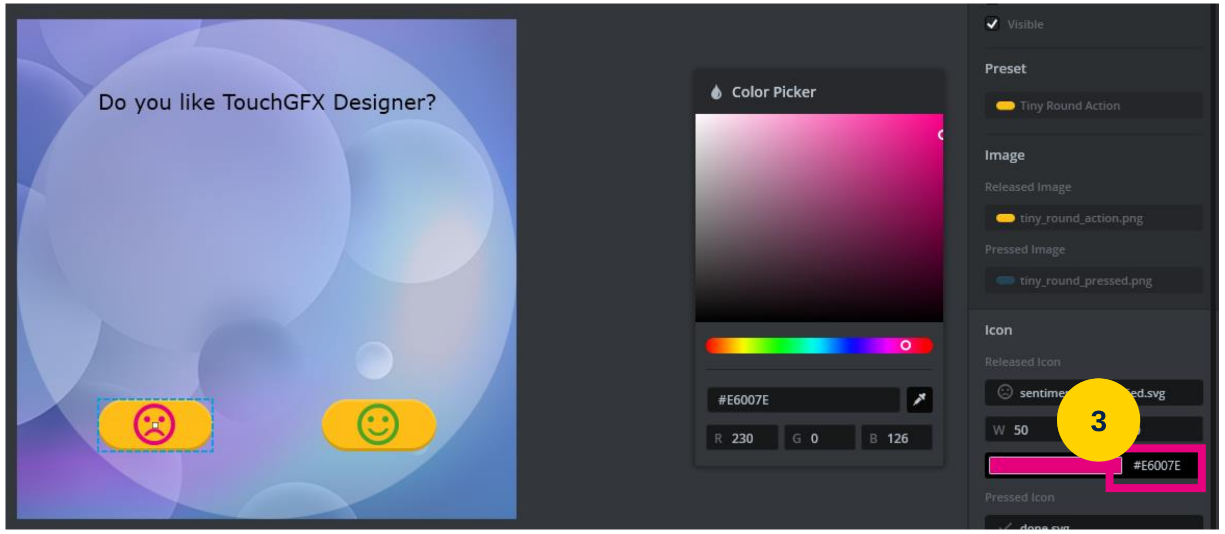


Setup button #2 (2/2)

1. Change default name “buttonHappy_1” to “buttonUnHappy” into Properties tab
2. Drag&Drop buttonUnHappy to be aside buttonHappy
3. Repeat steps 1…5 in slide #23, using:
   - **sentiment_dissatisfied.svg**
   - Another color than Green (e.g. **Pink**: #E6007E)



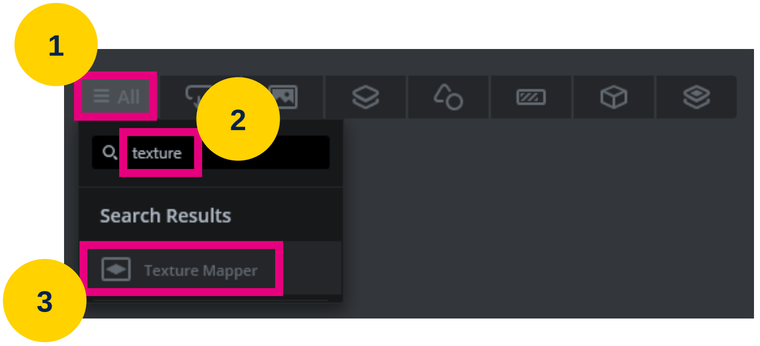
buttonUnHappy
X 78 Y 365
W 110 H 50
Tiny Round Action
tiny_round_action.png
tiny_round_pressed.png
sentiment_satisfied.svg
W 50 H 50

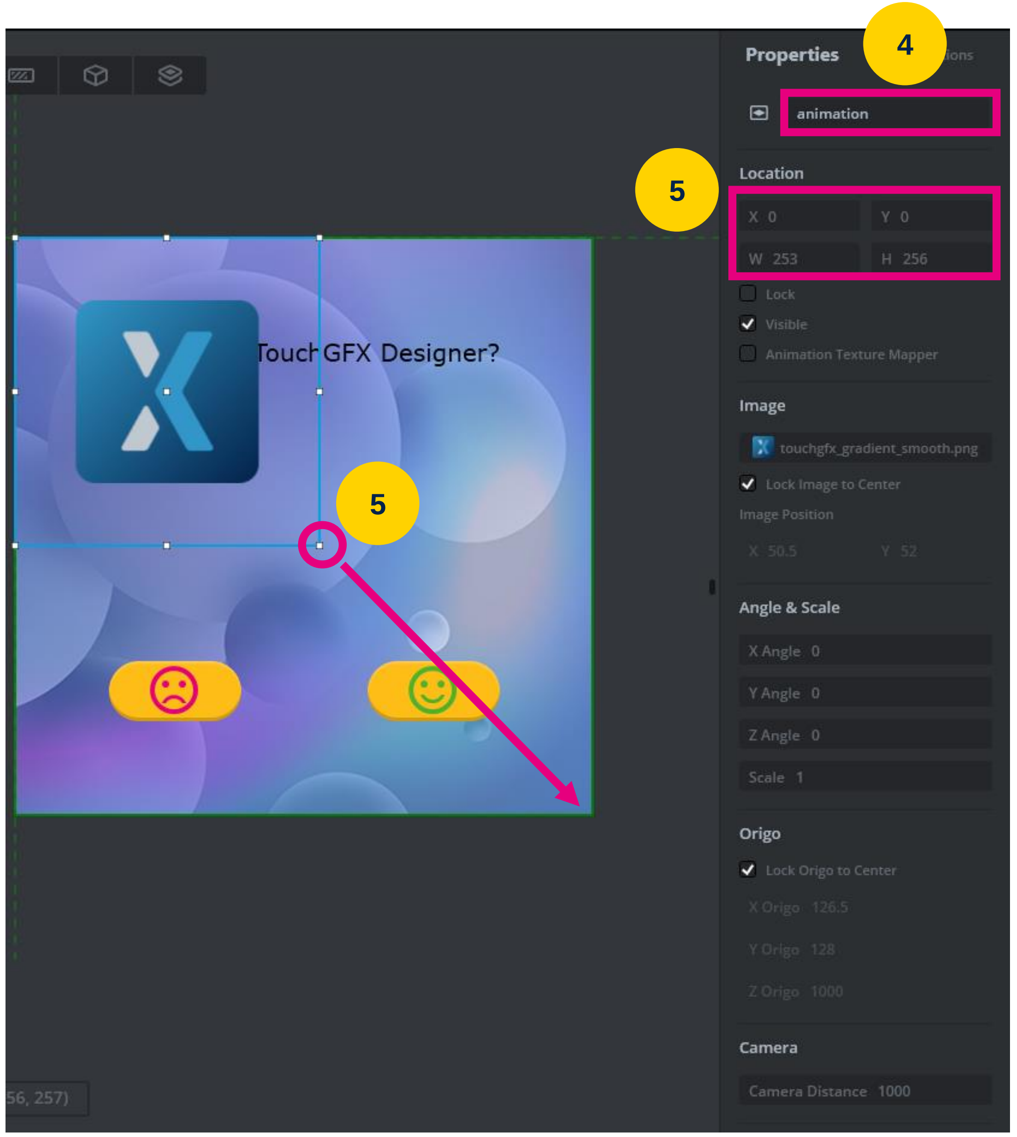


#E6007E
R 230 G 0 B 126
Tiny Round Action
tiny_round_action.png
tiny_round_pressed.png
sentime…ed.svg
W 50
#E6007E
done.svg

Setup texture mapper (1/4)

1. Mouse over "All" Widget Menu
2. Type "texture"
3. Select **Texture Mapper** on result
4. Change default name "textureMapper1" to "animation" into Properties tab
5. Drag the animation's corner to be full screen so that **Location**: X=0, Y=0, W=480, H=480

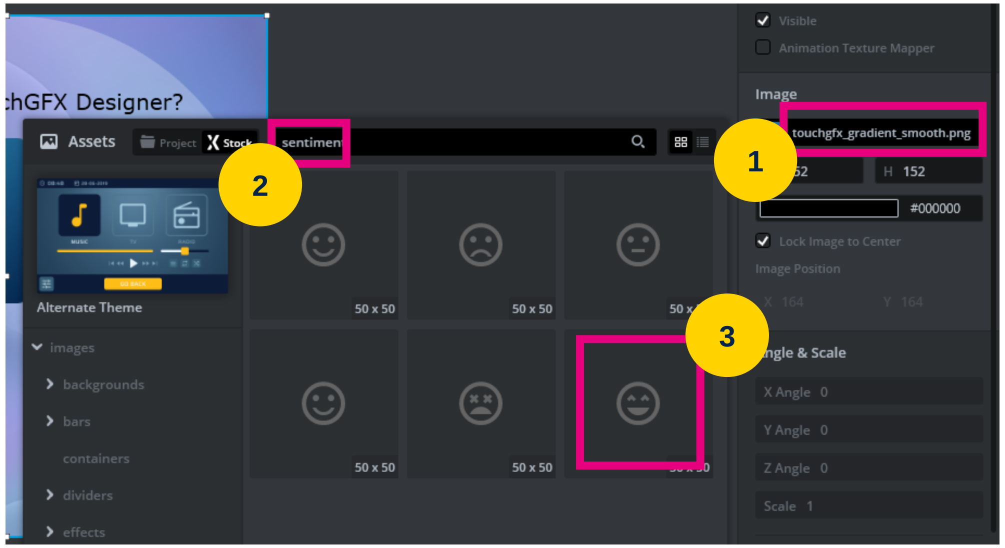


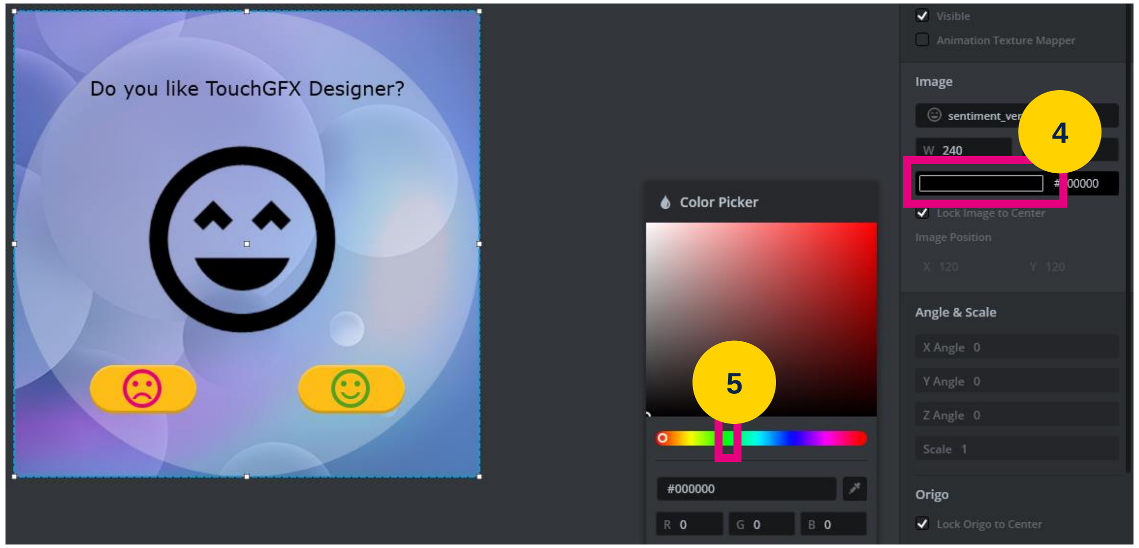


Properties
animation
Location
X 0 Y 0
W 253 H 256
Lock
Visible
Animation Texture Mapper
Image
touchgfx_gradient_smooth.png
Lock Image to Center
Image Position
X 50.5 Y 52
Angle & Scale
X Angle 0
Y Angle 0
Z Angle 0
Scale 1
Origo
Lock Origo to Center
X Origo 126.5
Y Origo 128
Z Origo 1000
Camera
Camera Distance 1000

Setup texture mapper (2/4)

1. Click on **touchgfx_gradient_smooth.png** in Image section of Properties tab
2. Type “sentiment”
3. Select sentiment_very_satisfied.svg (the big smile) on result
4. Click on *Color Picker* in Icon section of Properties tab
5. Select **Green** as color by mouse click *(in color bar first then in upper panel)*

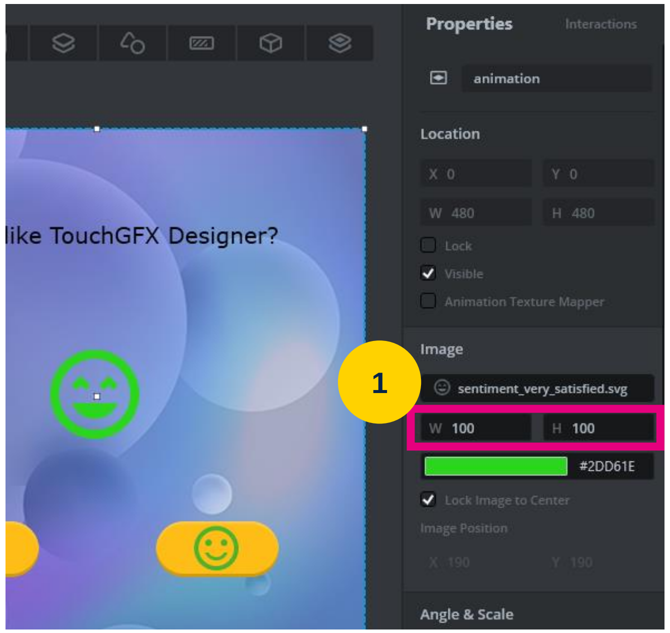




Setup texture mapper (3/4)

1
2
3
4

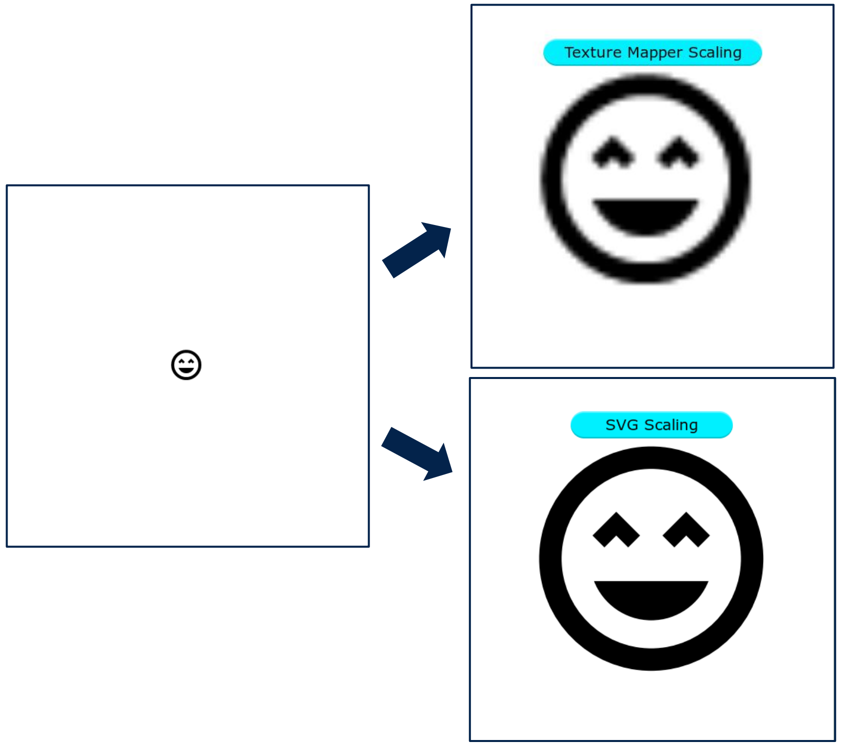
1. Set image size to
   W =100
   H =100



Even if we use an input SVG image, once in a texture mapper it will be handled as a bitmap.

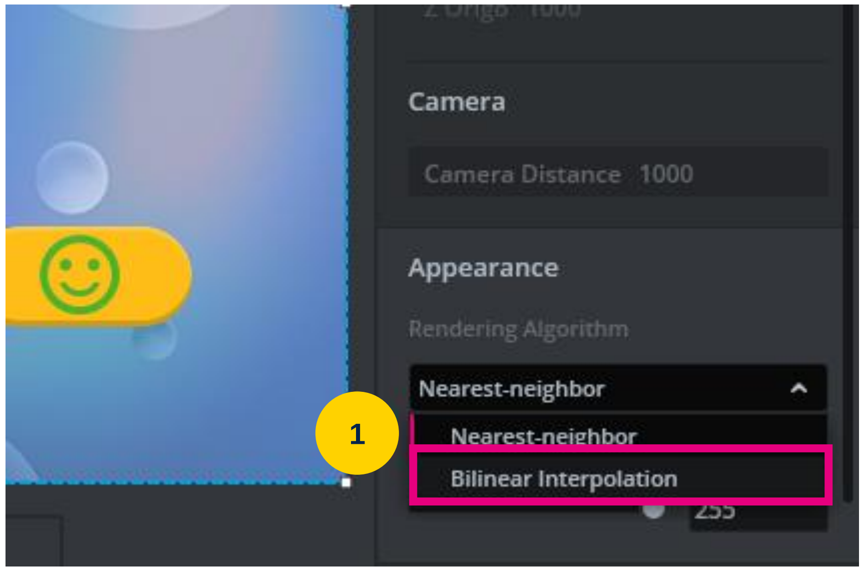
Upscaling a bitmap using a texture mapper introduces noise.
The larger the original image, the better the upscale will be.

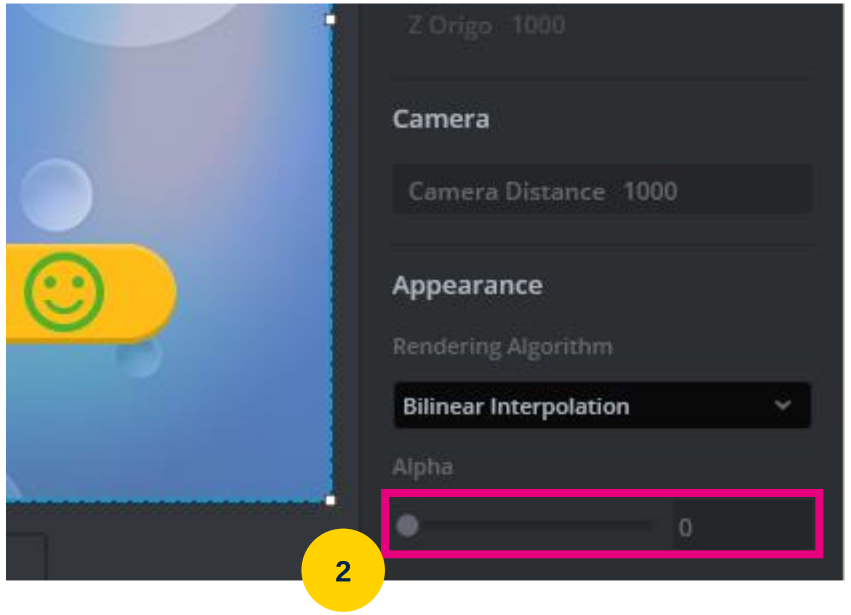
The exact same effect could be achieved using pure SVG element and transformation but not fully inside the Designer
(need to edit user code, less convenient for this training format)



Setup texture mapper (4/4)

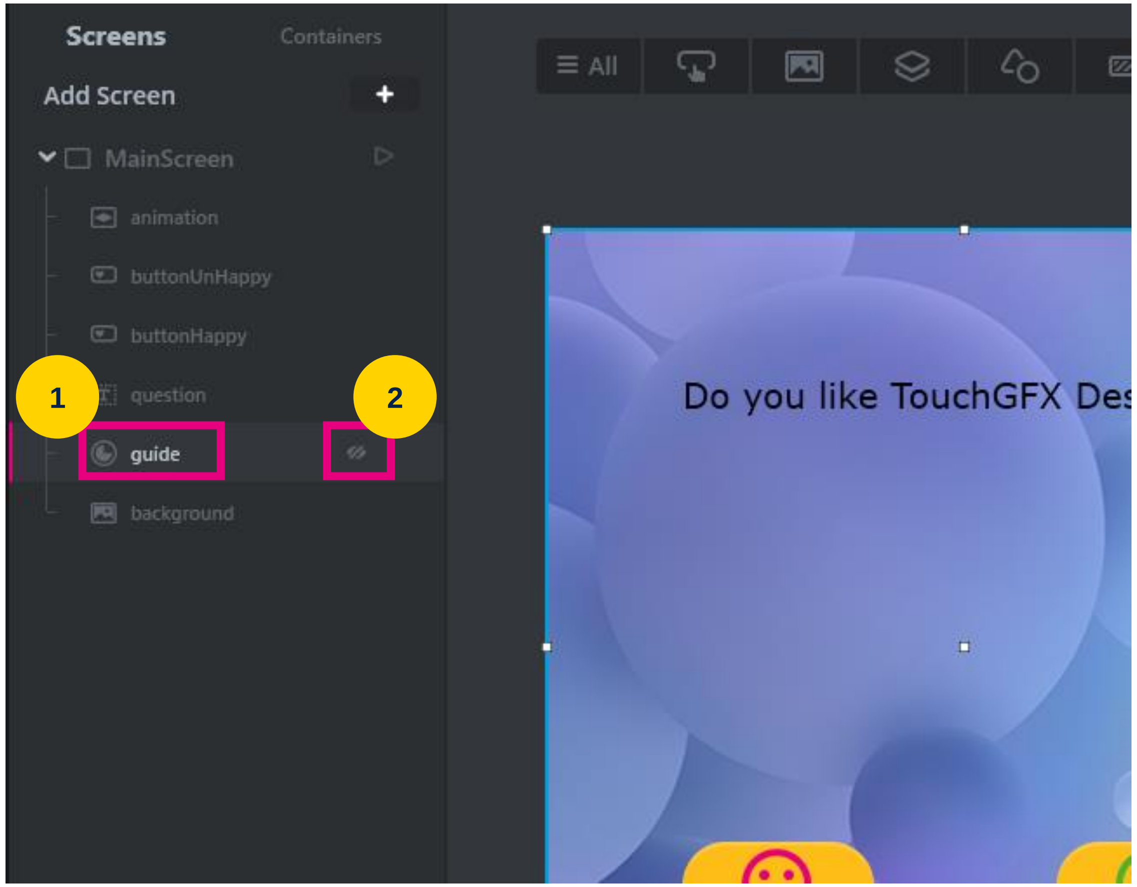
1. In *Appearance* section:
   click on “Bilinear Interpolation” to open menu
   select “Nearest-neighbor”
2. Set Alpha to 0 to make transparent


Z Origo 1000
Camera
Camera Distance 1000
Appearance
Rendering Algorithm
Nearest-neighbor
Nearest-neighbor
Bilinear Interpolation
255


Z Origo 1000
Camera
Camera Distance 1000
Appearance
Rendering Algorithm
Bilinear Interpolation
Alpha
0

Complete the setup

1. Select **guide** component
2. Click on eye icon to disable the guide circle (circle object remains in the screen)

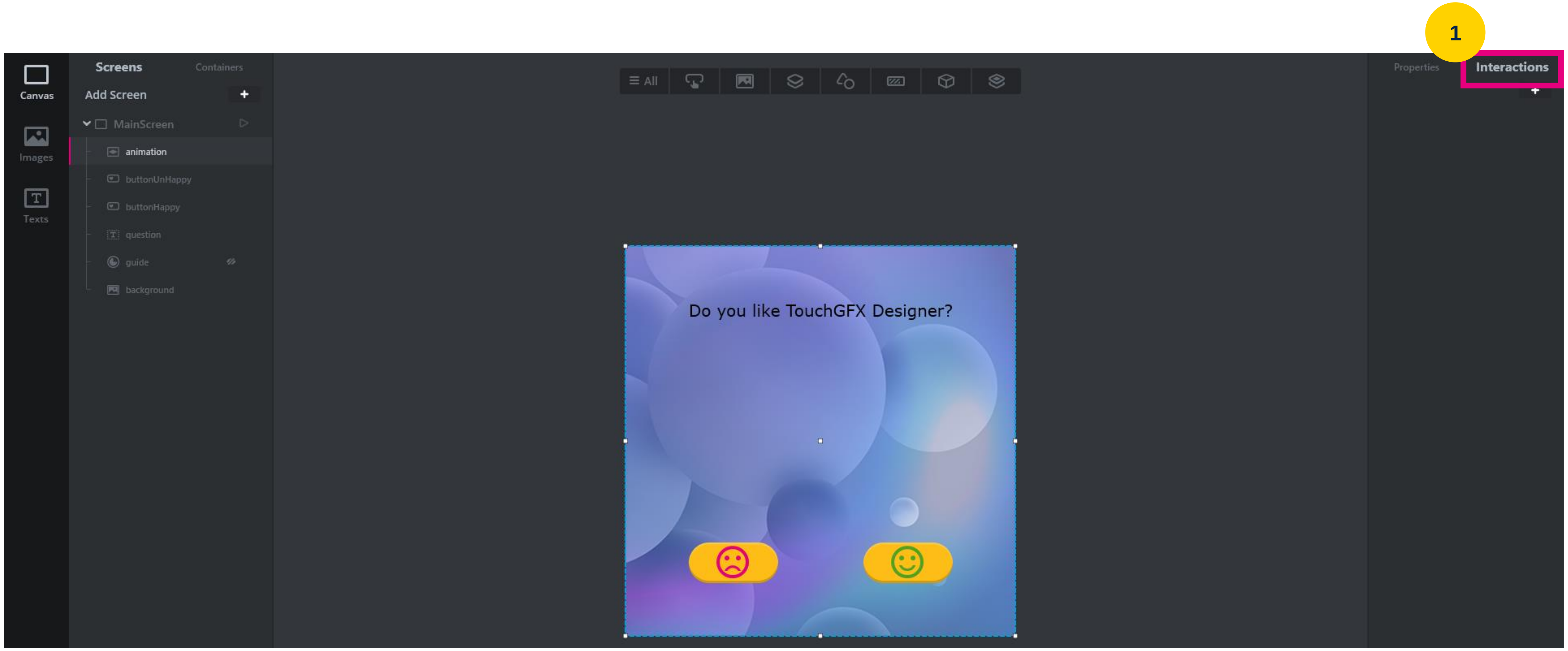


Setup the interactions

life.augmented

Where we are

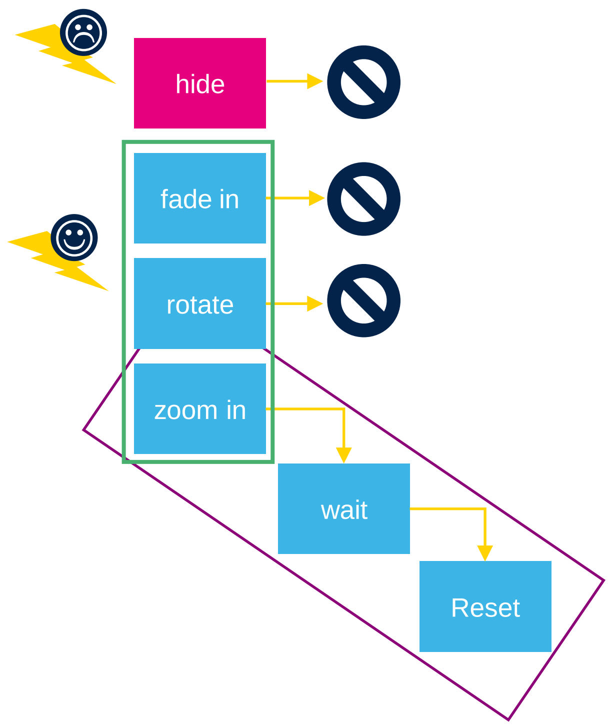
1. Click on **Interactions** tab



life.augmented

Think about the result at first

- We want to perform following macro steps:
  1. Hide **buttonUnHappy** when clicked
  2. Start animation when **buttonHappy** is clicked
  3. Wait and reset the screen
- How animation will work:
  - alpha (transparency) + rotation + scaling
    - In parallel
    - Trigger is buttonHappy click
  - Wait + reset screen
    - In series
    - Trigger is previous interaction done

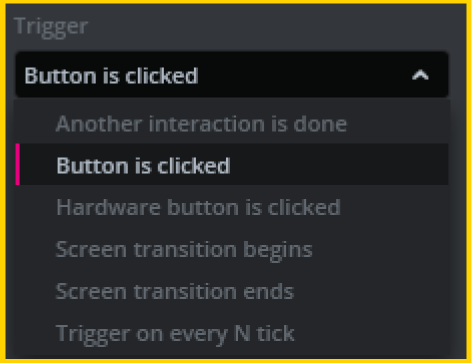


We'll set everything from the TouchGFX Designer: source, fx, duration, trigger, etc

Interaction is all about Trigger + Action

- Both Trigger and Actions dropdown are populated based on widgets of the selected Screen
  - E.g.: an empty Screen will only have 3 Triggers (Hardware button is clicked, Screen transition begins and Screen transition ends) and 5 Actions (Call new virtual function, Change screen, Execute C++ code, Set Language and Wait for
  - In our demo we are enabling a richer set of them, since using several widgets
- Some Triggers and Actions, e.g. “Button is clicked” and “Move widget”, require a component to be selected. However, if there is only one widget matching the Trigger/Action, that widget will be auto-selected
- Selecting some Actions, e.g. “Move widget”, also adds more properties relevant to moving a widget

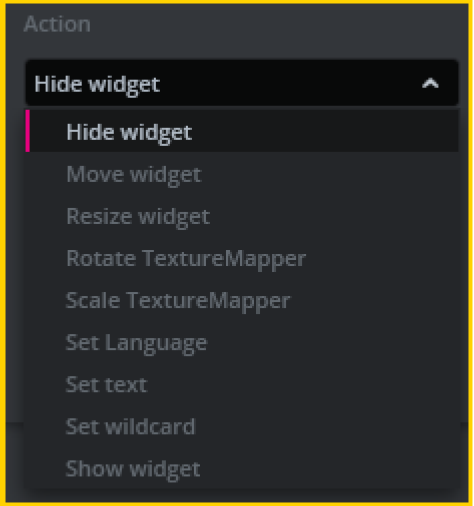
If


Trigger

Button is clicked

- Another interaction is done
- Button is clicked
- Hardware button is clicked
- Screen transition begins
- Screen transition ends
- Trigger on every N tick

Then


Action

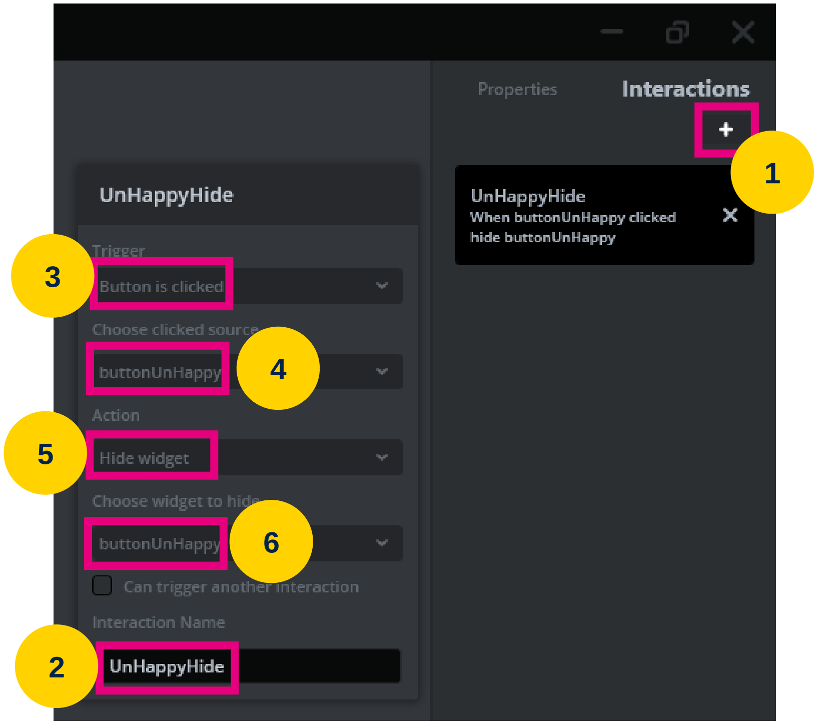
Hide widget

- Hide widget
- Move widget
- Resize widget
- Rotate TextureMapper
- Scale TextureMapper
- Set Language
- Set text
- Set wildcard
- Show widget

All interaction works like this: if TRIGGER then ACTION

1st interaction: hide a widget

1. Click on **+**
2. Change default **Interaction Name** to “UnHappyHide”
3. Setup **Trigger** to “Button is clicked”
4. Setup **Choose clicked source** to “buttonUnHappy”
5. Setup **Action** to “Hide widget”
6. Setup **Choose widget to hide** to “buttonUnHappy”



UnHappyHide

Trigger: Button is clicked

Choose clicked source: buttonUnHappy

Action: Hide widget

Choose widget to hide: buttonUnHappy

Can trigger another interaction

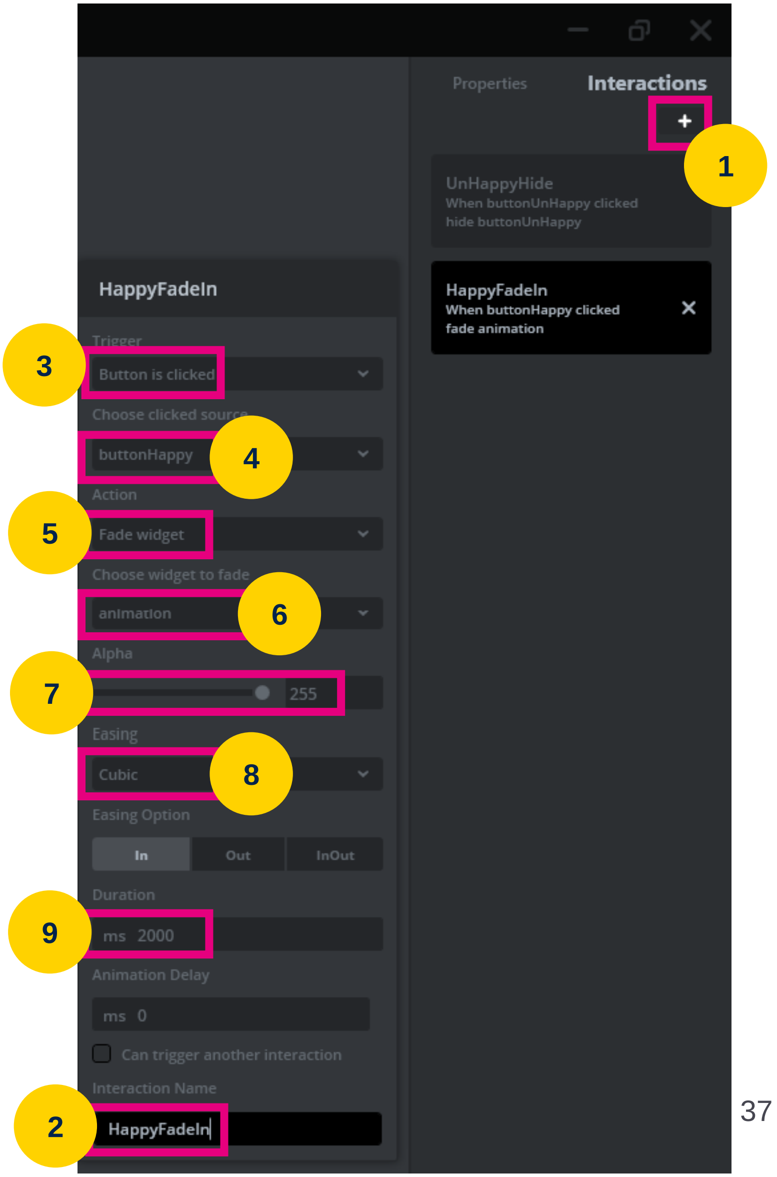
Interaction Name: UnHappyHide

UnHappyHide
When buttonUnHappy clicked
hide buttonUnHappy

Check and congrats: you added your first Interaction on TouchGFX Designer!

2nd interaction: make a widget slowly appearing

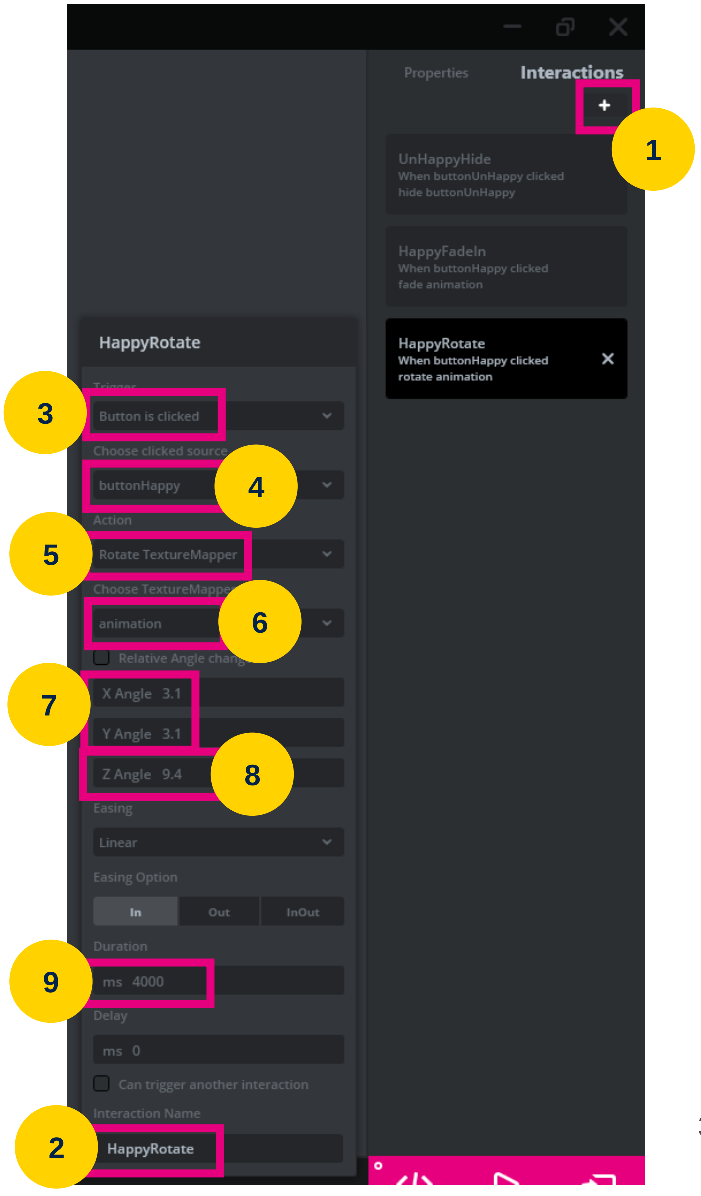
1. Click on **+**
2. Change default **Interaction Name** to “HappyFadeIn”
3. Setup **Trigger** to “Button is clicked”
4. Setup **Choose clicked source** to “buttonHappy”
5. Setup **Action** to “Fade widget”
6. Setup **Choose widget to fade** to “animation ”
7. Setup **Alpha** to **255** (make the *animation* widget visible)
8. Setup **Easing** to "Cubic"
9. Setup **Duration** to "2000"



This step will be repeated other 2 times (rotation + scaling)

3rd interaction: make a widget slowly rotating

1. Click on **+**
2. Change default **Interaction Name** to “HappyRotate”
3. Setup **Trigger** to “Button is clicked”
4. Setup **Choose clicked source** to “buttonHappy”
5. Setup **Action** to “Rotate TextureMapper ”
6. Setup **Choose TextureMapper** to “animation”
7. Setup both **X and Y Angles** to "3.1"
8. Setup **Z Angle** to "9.4"
9. Setup **Duration** to "4000"



UnHappyHide
When buttonUnHappy clicked
hide buttonUnHappy

HappyFadeIn
When buttonHappy clicked
fade animation

HappyRotate
When buttonHappy clicked
rotate animation

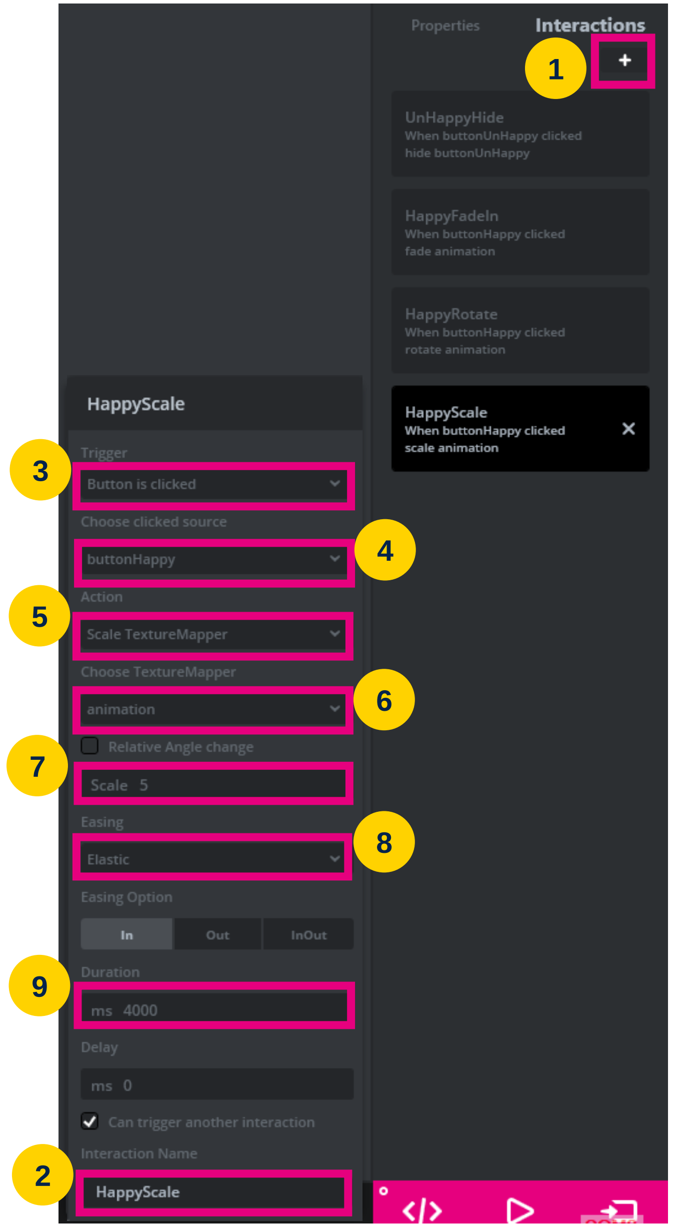
HappyRotate

Trigger: Button is clicked
Choose clicked source: buttonHappy
Action: Rotate TextureMapper
Choose TextureMapper: animation
Relative Angle chang
X Angle 3.1
Y Angle 3.1
Z Angle 9.4
Easing: Linear
Easing Option: [✓ In] Out InOut
Duration: ms 4000
Delay: ms 0
Can trigger another interaction
Interaction Name: HappyRotate

This step will be repeated one more time (scaling)

4th interaction: make a widget slowly zooming

1. Click on **+**
2. Change default **Interaction Name** to “HappyScale”
3. Setup **Trigger** to “Button is clicked”
4. Setup **Choose clicked source** to “buttonHappy”
5. Setup **Action** to “Scale TextureMapper”
6. Setup **Choose TextureMapper** to “animation”
7. Setup **Scale** to “5"
8. Setup **Easing** to "Elastic"
9. Setup **Duration** to "4000“



UnHappyHide
When buttonUnHappy clicked
hide buttonUnHappy

HappyFadeIn
When buttonHappy clicked
fade animation

HappyRotate
When buttonHappy clicked
rotate animation

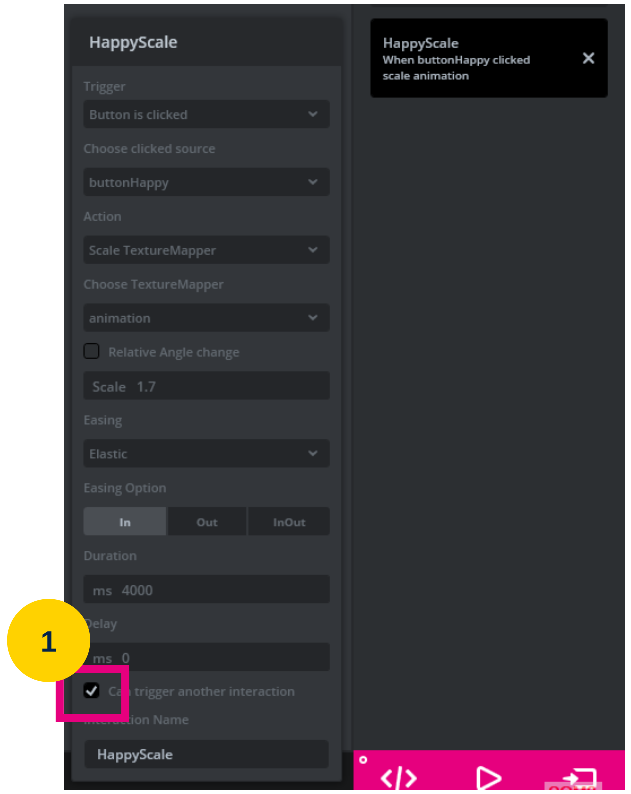
HappyScale
When buttonHappy clicked
scale animation

HappyScale

Trigger: Button is clicked
Choose clicked source: buttonHappy
Action: Scale TextureMapper
Choose TextureMapper: animation
Relative Angle change
Scale 5
Easing: Elastic
Easing Option: In | Out | InOut
Duration: ms 4000
Delay: ms 0
✓ Can trigger another interaction
Interaction Name: HappyScale

Where we are

1. Mark the “Can trigger another interaction” flag



HappyScale

Trigger: Button is clicked

Choose clicked source: buttonHappy

Action: Scale TextureMapper

Choose TextureMapper: animation

☐ Relative Angle change

Scale 1.7

Easing: Elastic

Easing Option: [✓ In] Out InOut

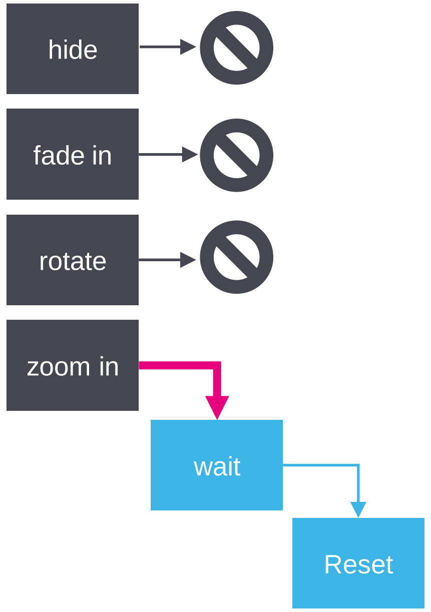
Duration: ms 4000

elay: ms 0

☑ Ca trigger another interaction

nteraction Name: HappyScale

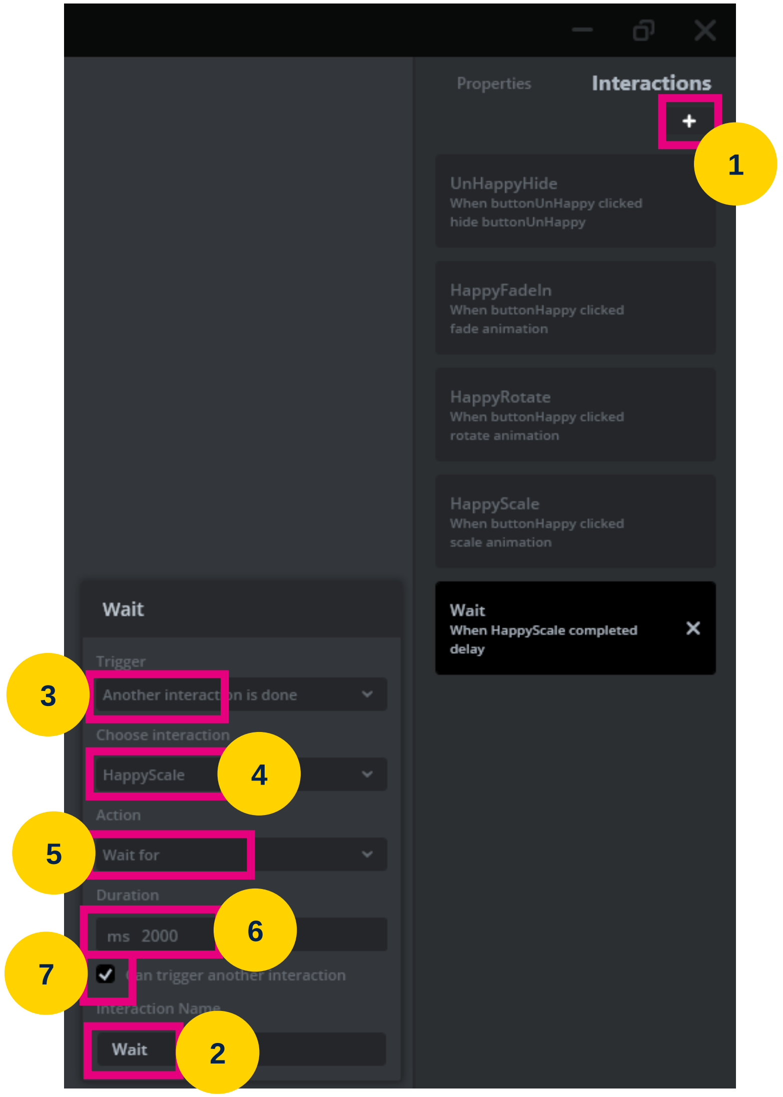
HappyScale
When buttonHappy clicked
scale animation



Let’s start now chaining the interactions

5th interaction: wait before resetting

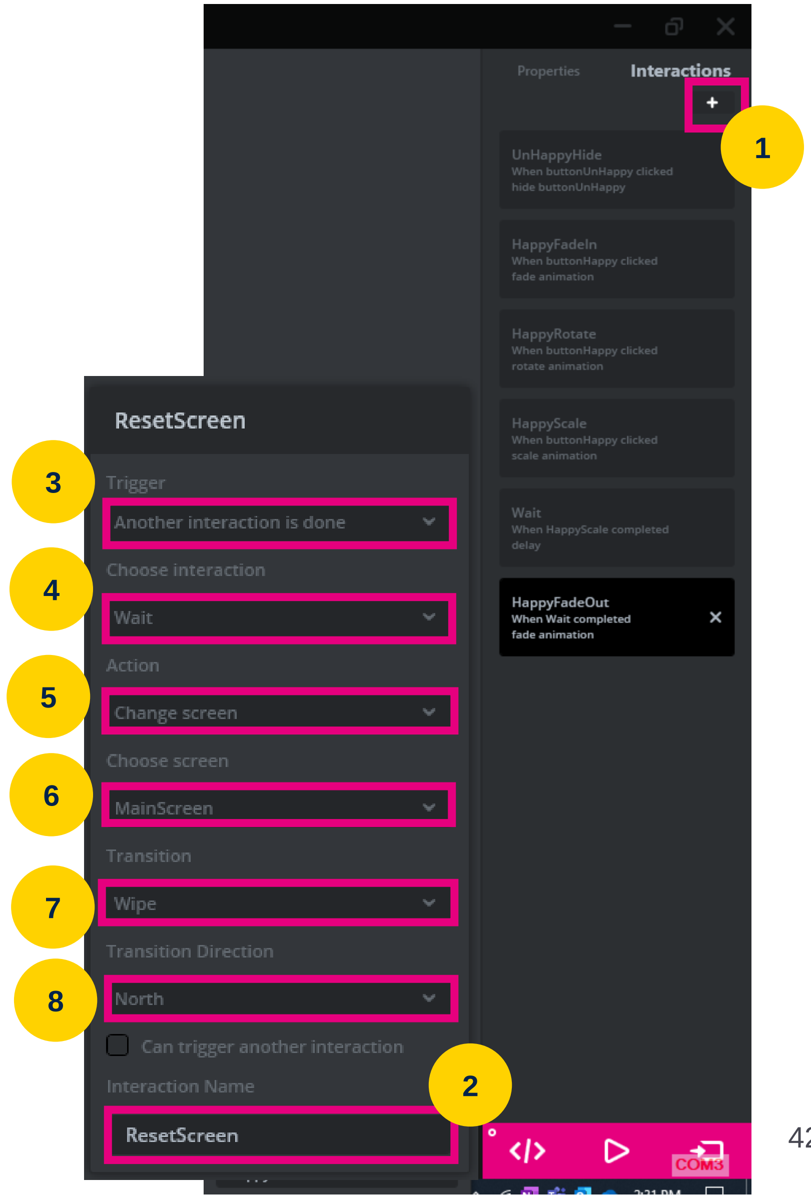
1. Click on **+**
2. Change default **Interaction Name** to "Wait"
3. Setup **Trigger** to "Another interaction is done"
4. Setup **Choose interaction** to "HappyScale"
5. Setup **Action** to "Wait for"
6. Setup **Duration** to "2000"
7. Mark the **Can trigger another interaction** flag



Trigger: Another interaction is done
Choose interaction: HappyScale
Action: Wait for
Duration: ms 2000
✓ Can trigger another interaction
Interaction Name: Wait

6th interaction: reset screen

1. Click on **+**
2. Change default **Interaction Name** to “ResetScreen”
3. Setup **Trigger** to “Another interaction is done”
4. Setup **Choose interaction** to “Wait”
5. Setup **Action** to “Change Screen”
6. Setup **Choose screen** to “MainScreen”
7. Setup **Transition** to “wipe"
8. Setup **Transition Direction** to “North"



UnHappyHide
When buttonUnHappy clicked
hide buttonUnHappy

HappyFadeIn
When buttonHappy clicked
fade animation

HappyRotate
When buttonHappy clicked
rotate animation

HappyScale
When buttonHappy clicked
scale animation

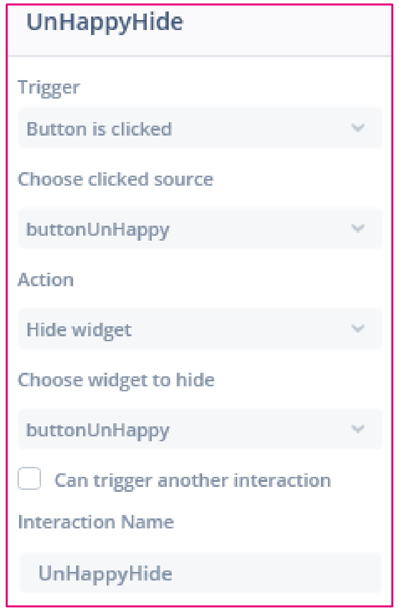
Wait
When HappyScale completed
delay

HappyFadeOut
When Wait completed
fade animation

ResetScreen

Trigger: Another interaction is done
Choose interaction: Wait
Action: Change screen
Choose screen: MainScreen
Transition: Wipe
Transition Direction: North
Can trigger another interaction
Interaction Name: ResetScreen

Interactions summary


UnHappyHide

Trigger
Button is clicked

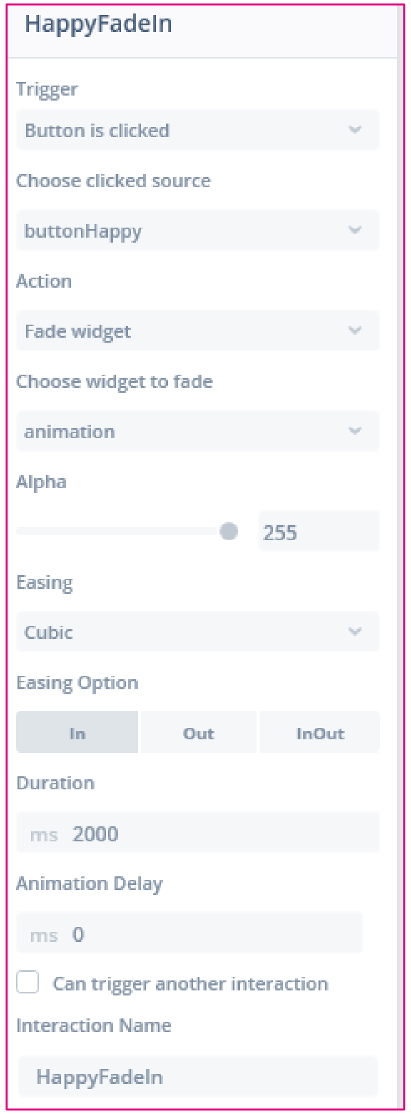
Choose clicked source
buttonUnHappy

Action
Hide widget

Choose widget to hide
buttonUnHappy

☐ Can trigger another interaction

Interaction Name
UnHappyHide


HappyFadeIn

Trigger
Button is clicked

Choose clicked source
buttonHappy

Action
Fade widget

Choose widget to fade
animation

Alpha
255

Easing
Cubic

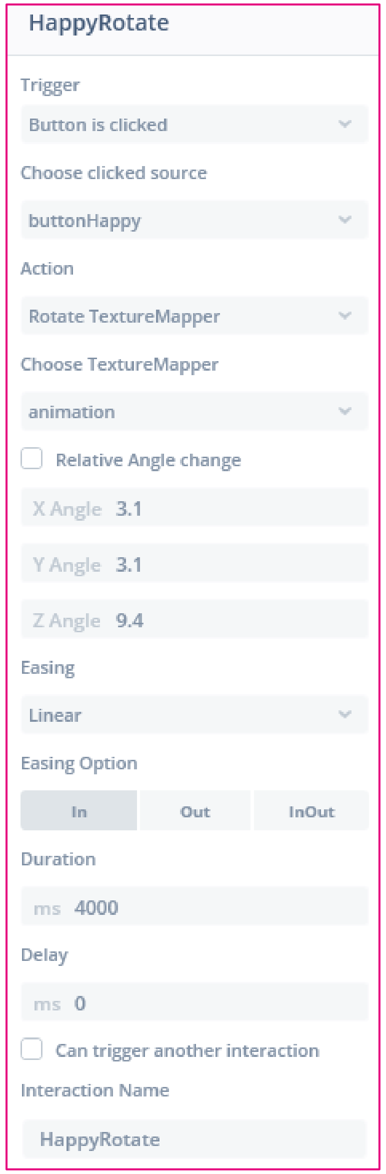
Easing Option
[✓ In] Out InOut

Duration
ms 2000

Animation Delay
ms 0

☐ Can trigger another interaction

Interaction Name
HappyFadeIn


HappyRotate

Trigger
Button is clicked

Choose clicked source
buttonHappy

Action
Rotate TextureMapper

Choose TextureMapper
animation

☐ Relative Angle change

X Angle 3.1

Y Angle 3.1

Z Angle 9.4

Easing
Linear

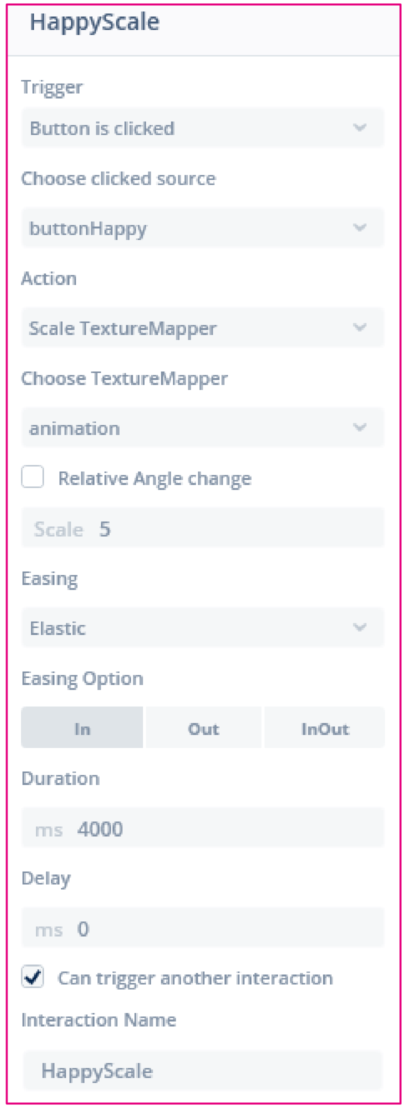
Easing Option
[✓ In] Out InOut

Duration
ms 4000

Delay
ms 0

☐ Can trigger another interaction

Interaction Name
HappyRotate


HappyScale

Trigger
Button is clicked

Choose clicked source
buttonHappy

Action
Scale TextureMapper

Choose TextureMapper
animation

☐ Relative Angle change

Scale 5

Easing
Elastic

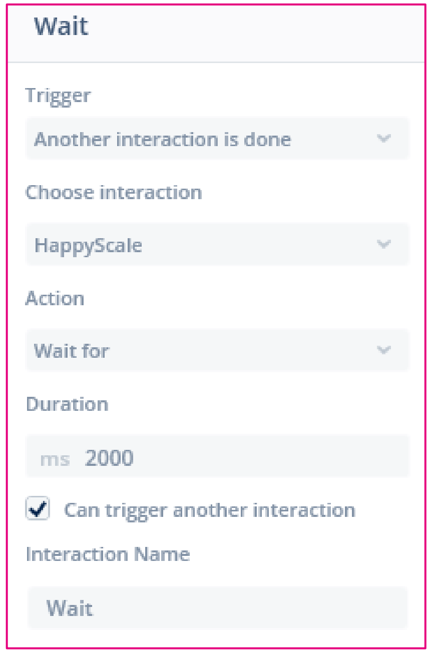
Easing Option
[✓ In] Out InOut

Duration
ms 4000

Delay
ms 0

☑ Can trigger another interaction

Interaction Name
HappyScale


Wait

Trigger
Another interaction is done

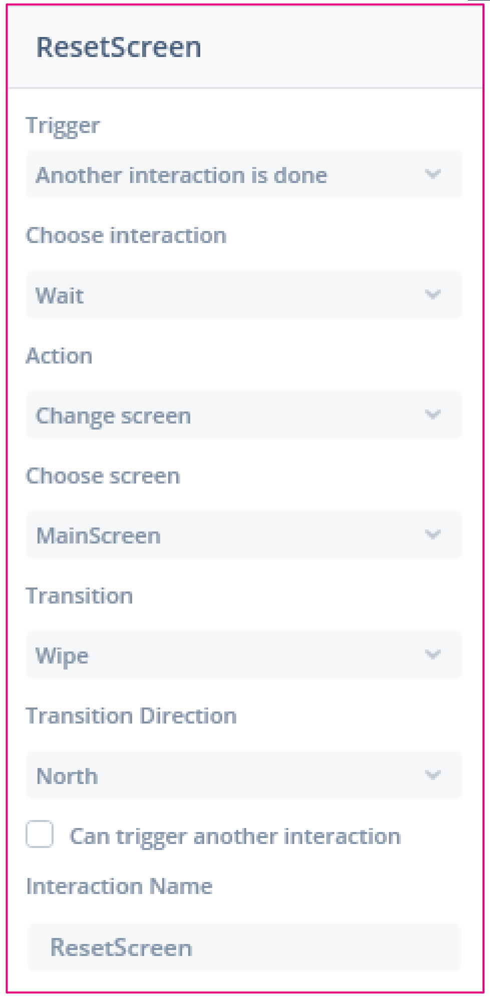
Choose interaction
HappyScale

Action
Wait for

Duration
ms 2000

☑ Can trigger another interaction

Interaction Name
Wait


ResetScreen

Trigger
Another interaction is done

Choose interaction
Wait

Action
Change screen

Choose screen
MainScreen

Transition
Wipe

Transition Direction
North

☐ Can trigger another interaction

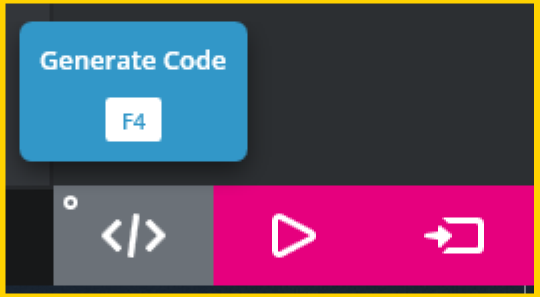
Interaction Name
ResetScreen

Compile and Run

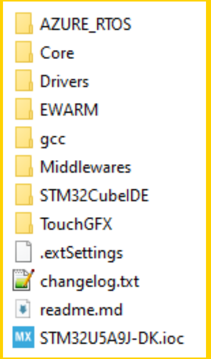
life.augmented

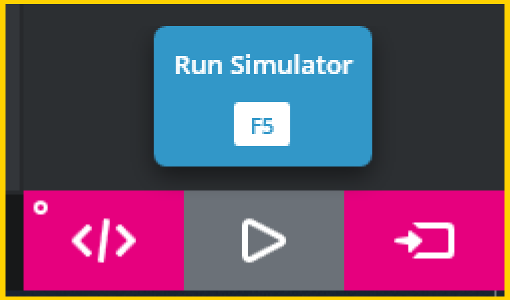
TouchGFX Designer: one tool to include them all

- We’ve just implemented our demo in wysiwyg way by TouchGFX Designer
- But… there is more than this in here.

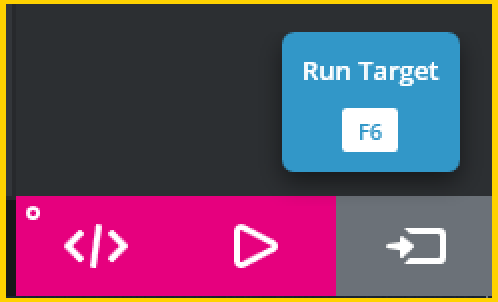


Code generation will create c++ files for defined screens update project (Keil, IAR or STM32CubeIDE)based on STM32CubeMX project


AZURE_RTOS
Core
Drivers
EWARM
gcc
Middlewares
STM32CubeIDE
TouchGFX
.extSettings
changelog.txt
readme.md
STM32U5A9J-DK.ioc



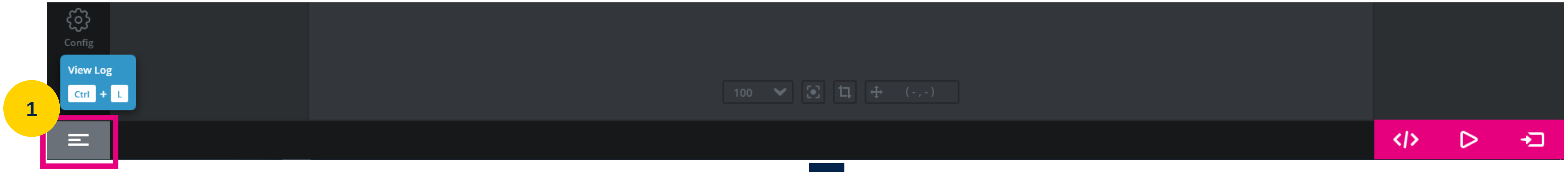
Code simulation is a great addition to your development tool. The code executed is exactly the same for the GUI as on target HW. hardware resources are the PC host’s

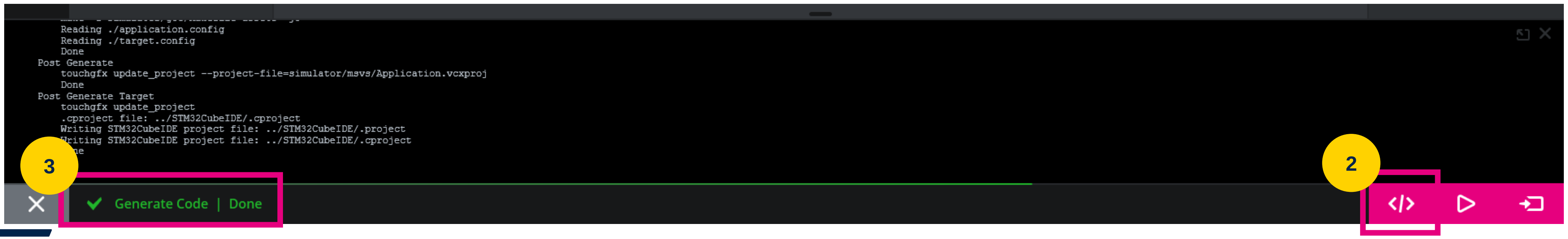


Embedded gcc toolchain will build the code for the target and STM32CubeProgrammer will be launched through this shortcut. You don’t need to open it externally to be able to flash the target board

Generate the project

1. Click on “View Log” button
2. Click on “Generate Code” button
3. Check the result, and ensure no errors are in there

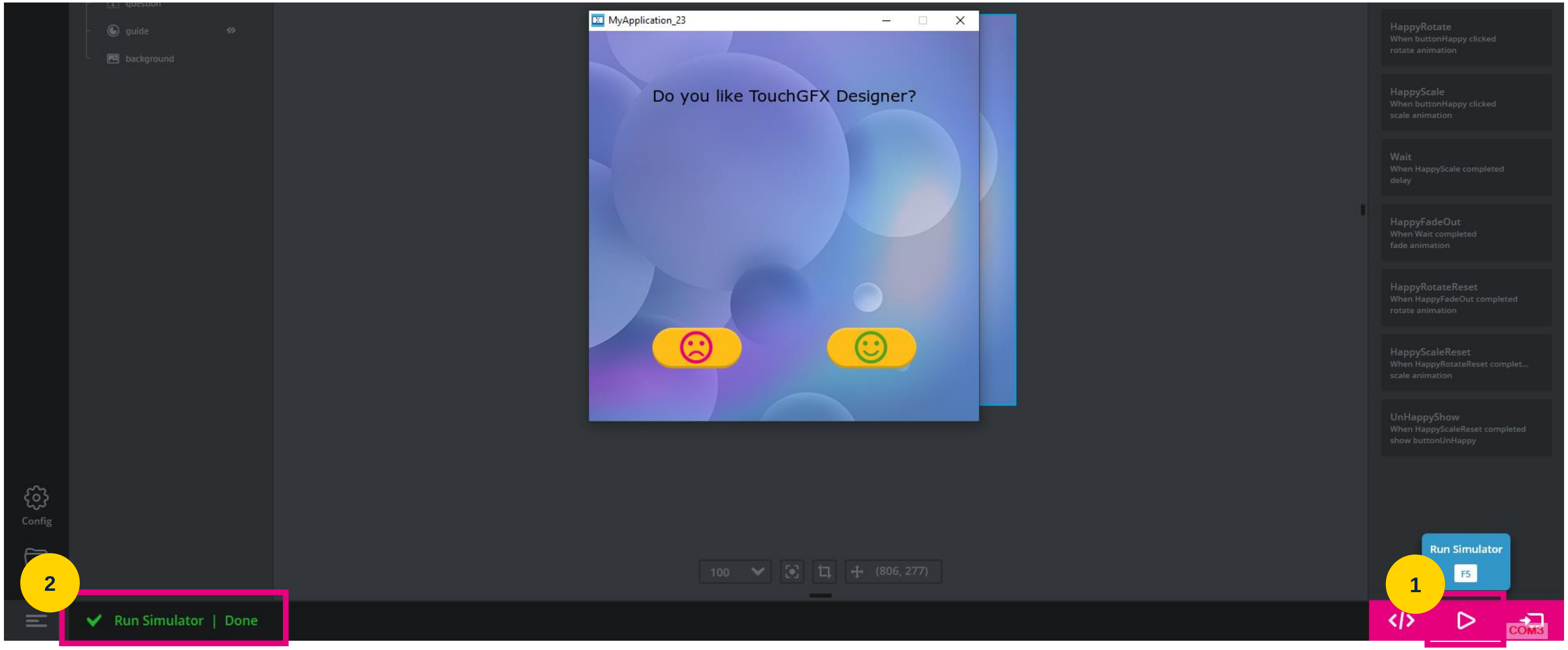




```
        Reading ./application.config
        Reading ./target.config
        Done
    Post Generate
        touchgfx update_project --project-file=simulator/msvs/Application.vcxproj
        Done
    Post Generate Target
        touchgfx update_project
        .cproject file: ../STM32CubeIDE/.cproject
        Writing STM32CubeIDE project file: ../STM32CubeIDE/.project
        Writing STM32CubeIDE project file: ../STM32CubeIDE/.cproject
        Done
```

Simulate the project

1. Click on “Run Simulator” button
2. Check the result, and ensure no errors are in there



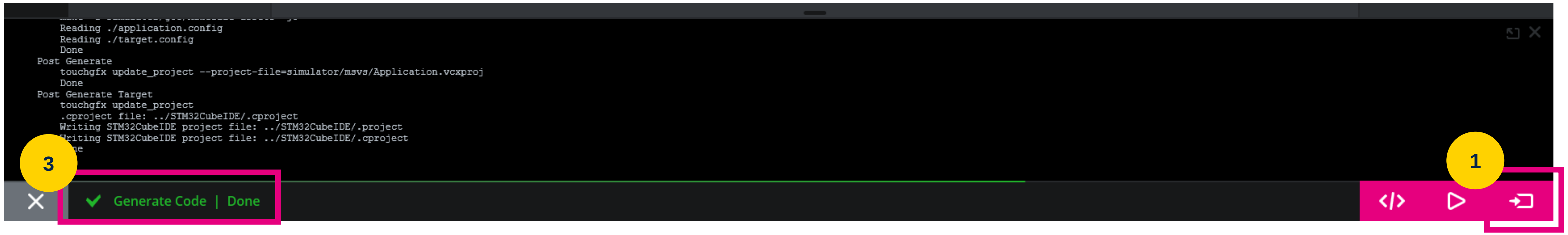
TouchGFX Simulator features

- Apart from capturing mouse input, the TouchGFX Simulator allows much more!
- Once in run mode, use your keyboard to test the animation (F9 → F10$^+$ → ESC)

Shortcut	Feature
F1	Enables/disables display of pointer coordinates as well as RGB color of the pixel at that coordinate
F2	Enables/disables highlighting invalidated area
F3	Takes a screenshot and places the image under the screenshots folder on Disk
Ctrl+F3	Takes screenshots of the next 50 frames and places the images under the screenshots folder on Disk
Shift+F3	Takes a screenshot and places it in your clipboard
F4	If a simulator skin is used → enables/disables the simulator skin If a simulator skin is not used → enables/disables window border
F5	Sends the application straight back to the startup screen by calling FrontendApplication::changeToStartScreen()
F9	Pauses/Resumes the simulator by preventing ticks to be sent to the application
F10	While the simulator is paused (by F9) → send a single tick to the application ("single step")
ESC	Close the simulator

Flash the project

1. Click on “Run Target” button
2. Check the result, and ensure no errors are in there



```
        Reading ./application.config
        Reading ./target.config
        Done
    Post Generate
        touchgfx update_project --project-file=simulator/msvs/Application.vcxproj
        Done
    Post Generate Target
        touchgfx update_project
        .cproject file: ../STM32CubeIDE/.cproject
        Writing STM32CubeIDE project file: ../STM32CubeIDE/.project
        Writing STM32CubeIDE project file: ../STM32CubeIDE/.cproject
        Done
```

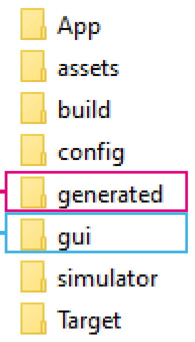
Check and congrats: your project has been now deployed on your STM32U599J-DK!

Move to C++ world

life.augmented

Let’s move to the real life

- The code generated by TouchGFX Designer will be completely separate from the code written by the user:
  - The generated code is placed in the folder **generated**/gui_generated
  - The handwritten code is placed in the **gui** folder



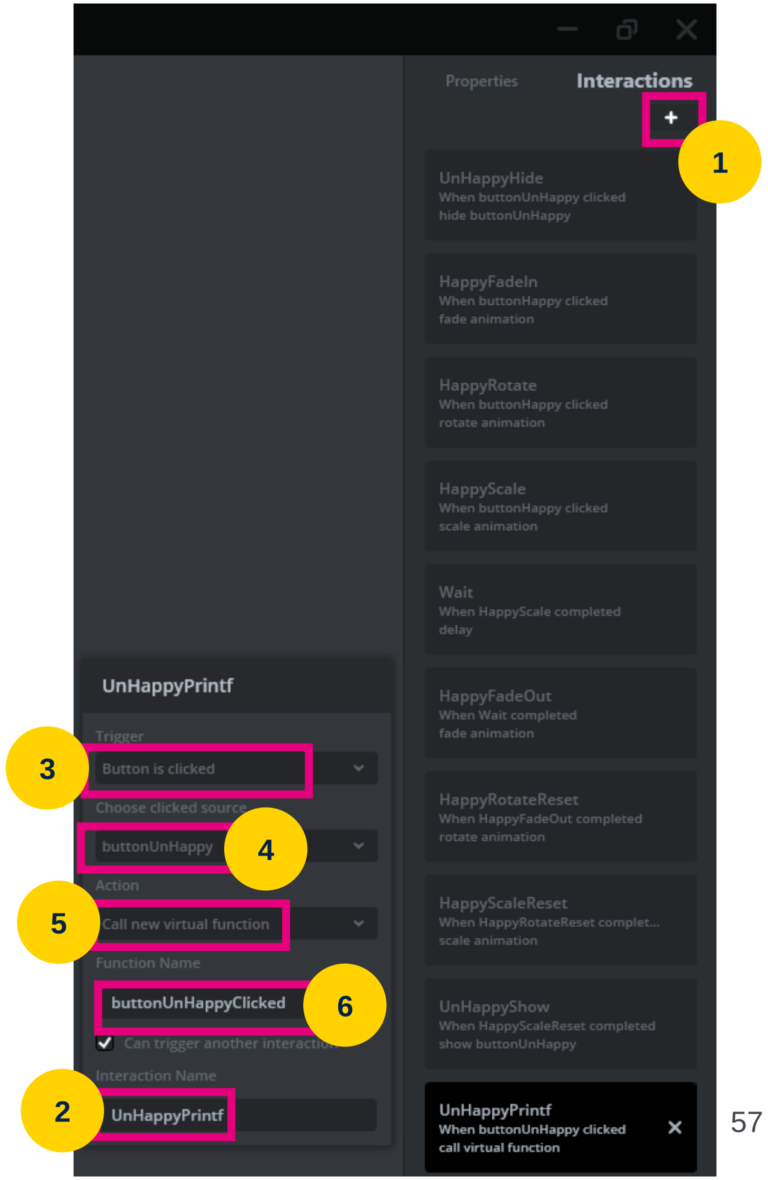
- **"generated" folder** and corresponding files contains **Base classes** of what is defined in the Designer UI. These files are overwritten at each code generation from the Designer. These files **should not be edited manually**.
- **User classes** are subclasses of the base ones so that users can implement custom features that are not supported in the Designer. The user classes will be generated only once and **will never be altered by TouchGFX Designer**.

We’ll add a simple printf, just to add some C++ code into user’s code

Call a C++ function

1. Click on **+**
2. Change default **Interaction Name** to “UnHappyPrintf”
3. Setup **Trigger** to “Button is clicked”
4. Setup **Choose clicked source** to “buttonUnHappy”
5. Setup **Action** to “Call new virtual function”
6. Setup **Function Name** to “buttonUnHappyClicked”

This will add an empty buttonUnHappyClicked() method to **MainScreenViewBase** class (generated), we will then overload it in the user code subclass **MainScreenView** to do what we want.



UnHappyPrintf
Trigger
Button is clicked
Choose clicked source
buttonUnHappy
Action
Call new virtual function
Function Name
buttonUnHappyClicked
✓ Can trigger another interaction
Interaction Name
UnHappyPrintf

UnHappyHide
When buttonUnHappy clicked
hide buttonUnHappy

HappyFadeIn
When buttonHappy clicked
fade animation

HappyRotate
When buttonHappy clicked
rotate animation

HappyScale
When buttonHappy clicked
scale animation

Wait
When HappyScale completed
delay

HappyFadeOut
When Wait completed
fade animation

HappyRotateReset
When HappyFadeOut completed
rotate animation

HappyScaleReset
When HappyRotateReset complet...
scale animation

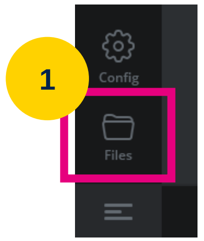
UnHappyShow
When HappyScaleReset completed
show buttonUnHappy

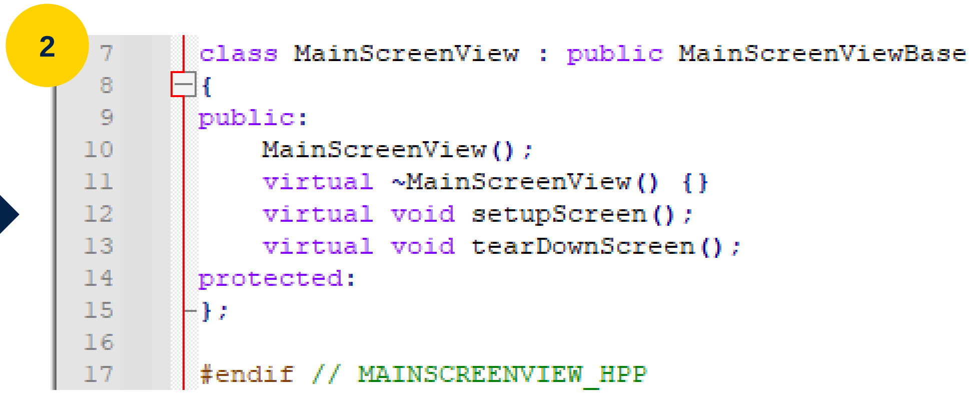
UnHappyPrintf
When buttonUnHappy clicked
call virtual function

Implement a C++ function (1/3)

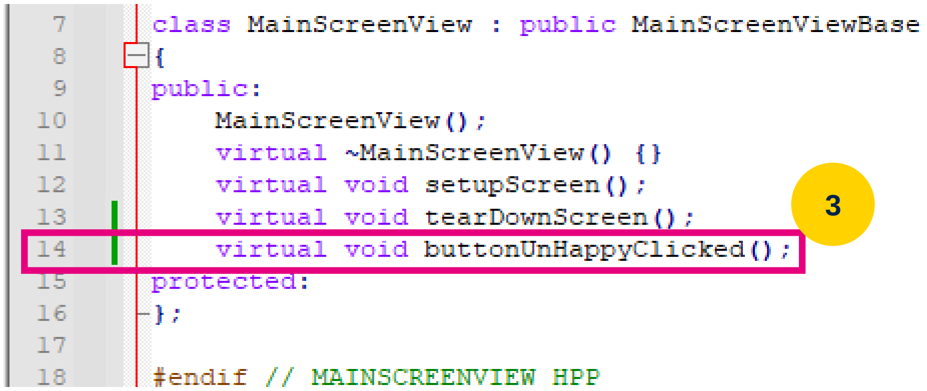
1. Click on **Files** button
2. Browse to “\gui\include\gui\mainscreen_screen” and open “MainScreenView.hpp”
3. Add code inside at line 14

```
virtual void buttonUnHappyClicked();
```





```
class MainScreenView : public MainScreenViewBase
{
public:
    MainScreenView();
    virtual ~MainScreenView() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
protected:
};

#endif // MAINSCREENVIEW_HPP
```



```
class MainScreenView : public MainScreenViewBase
{
public:
    MainScreenView();
    virtual ~MainScreenView() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
    virtual void buttonUnHappyClicked();
protected:
};

#endif // MAINSCREENVIEW_HPP
```

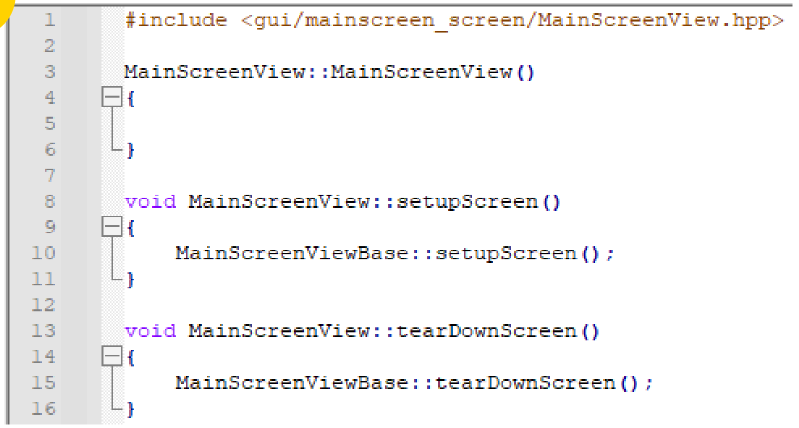
Implement a C++ function (Simulator) (2/3)

1. Browse to “\gui\src\gui\mainscreen_screen” and open “MainScreenView.cpp”
2. Add code inside at line 2
3. Add code inside at line 19
4. Simulate the project (Cf. Slide #46)

```
#include <touchgfx/utils.hpp>
```

```
void MainScreenView::buttonUnHappyClicked()
{
    touchgfx_printf("buttonUnHappyClicked\n");
}
```

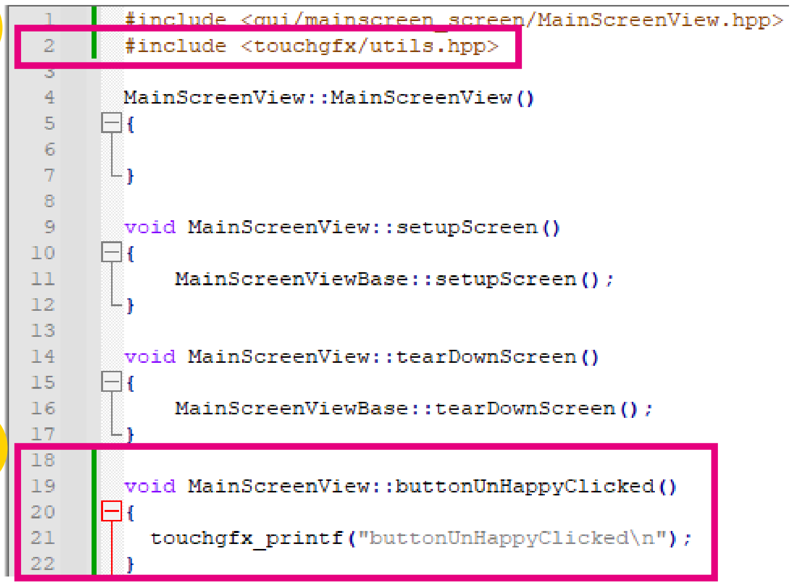
1



```
#include <gui/mainscreen_screen/MainScreenView.hpp>

MainScreenView::MainScreenView()
{

}

void MainScreenView::setupScreen()
{
    MainScreenViewBase::setupScreen();
}

void MainScreenView::tearDownScreen()
{
    MainScreenViewBase::tearDownScreen();
}
```

2

3



```
#include <gui/mainscreen_screen/MainScreenView.hpp>
#include <touchgfx/utils.hpp>

MainScreenView::MainScreenView()
{

}

void MainScreenView::setupScreen()
{
    MainScreenViewBase::setupScreen();
}

void MainScreenView::tearDownScreen()
{
    MainScreenViewBase::tearDownScreen();
}

void MainScreenView::buttonUnHappyClicked()
{
  touchgfx_printf("buttonUnHappyClicked\n");
}
```

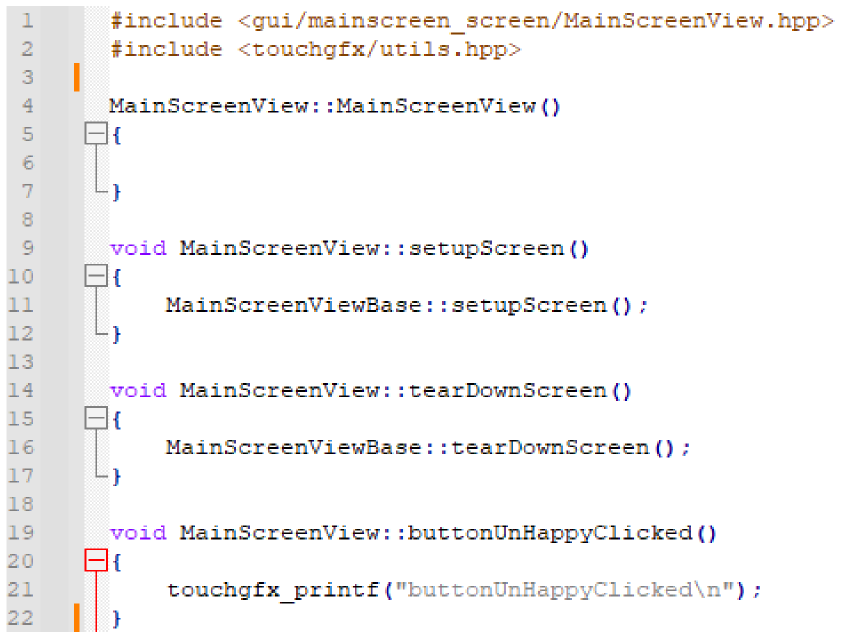
life.augmented

Implement a C++ function (Target) (3/3)

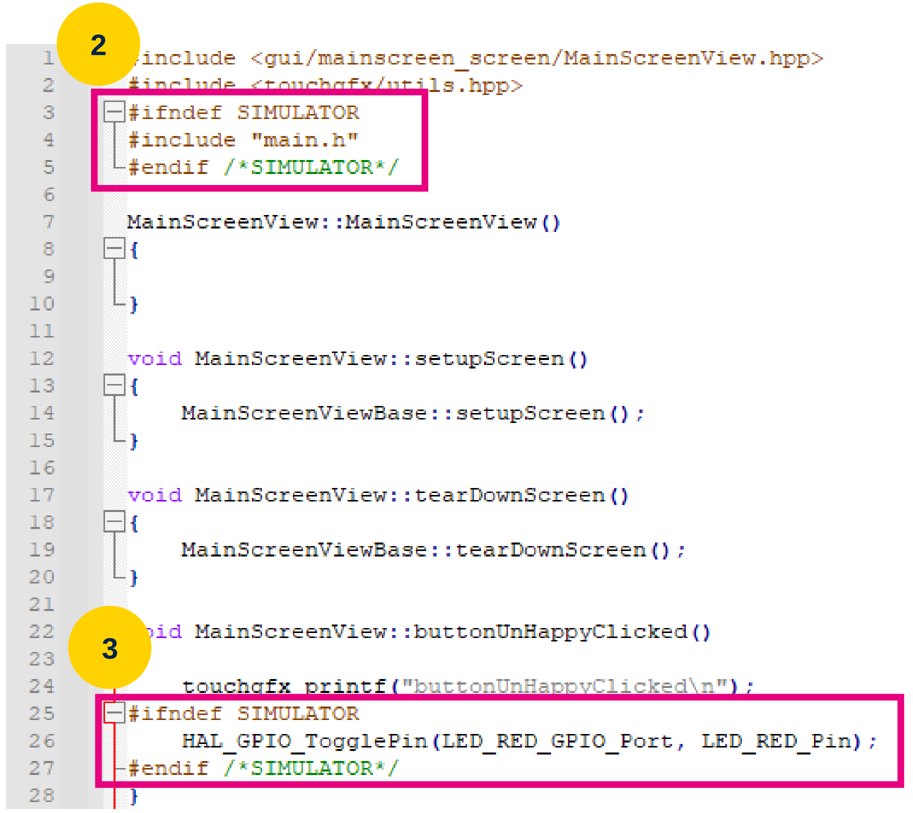
1. Add code inside at line 3
2. Add code inside at line 25
3. Simulate the project (Cfr Slide #46)

```
#ifndef SIMULATOR
#include "main.h“
#endif /*SIMULATOR*/
```

```
#ifndef SIMULATOR
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
#endif /*SIMULATOR*/
```



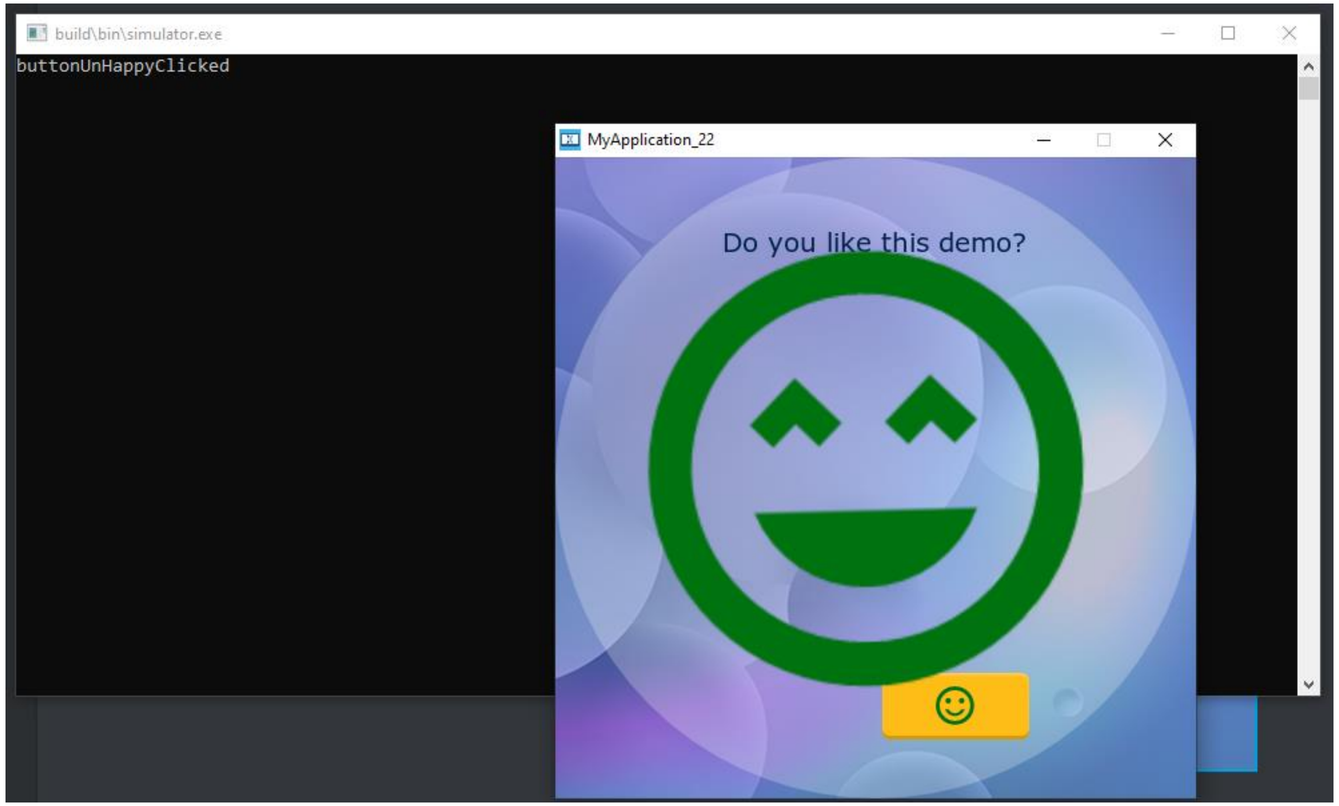
```
1   #include <gui/mainscreen_screen/MainScreenView.hpp>
2   #include <touchgfx/utils.hpp>
3
4   MainScreenView::MainScreenView()
5   {
6
7   }
8
9   void MainScreenView::setupScreen()
10  {
11      MainScreenViewBase::setupScreen();
12  }
13
14  void MainScreenView::tearDownScreen()
15  {
16      MainScreenViewBase::tearDownScreen();
17  }
18
19  void MainScreenView::buttonUnHappyClicked()
20  {
21      touchgfx_printf("buttonUnHappyClicked\n");
22  }
```



```
1   #include <gui/mainscreen_screen/MainScreenView.hpp>
2   #include <touchgfx/utils.hpp>
3   #ifndef SIMULATOR
4   #include "main.h"
5   #endif /*SIMULATOR*/
6
7   MainScreenView::MainScreenView()
8   {
9
10  }
11
12  void MainScreenView::setupScreen()
13  {
14      MainScreenViewBase::setupScreen();
15  }
16
17  void MainScreenView::tearDownScreen()
18  {
19      MainScreenViewBase::tearDownScreen();
20  }
21
22  void MainScreenView::buttonUnHappyClicked()
23  {
24      touchgfx_printf("buttonUnHappyClicked\n");
25  #ifndef SIMULATOR
26      HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
27  #endif /*SIMULATOR*/
28  }
```

The result of the workshop

- Click to buttonHappy to start the animation (no changes)
- Click to buttonUnHappy to force the printf (new)



```
buttonUnHappyClicked
```

Check and congrats: you are now a TouchGFX user!

Takeaways

life.augmented

Take-aways

The best GUI Solution in the market	STM32 MCU + TouchGFX gives you high performance UIs on market proven hardware
Easy & Fast prototyping	With ONLY 5 clicks you can flash a TouchGFX demo on an STM32 DISCO
Largest MCU portfolio for running GUIs	You can run TouchGFX from STM32G0 up to STM32H7s and all in between
STM32U5x9 for high integration and simple PCB	Reach lower BOM by not needing any external RAM for placing framebuffers
NeoChrom GPU for advanced GFX operations	Get better UI performance with less resources and less power consumption

Amaze your customers with the easiness TouchGFX on STM32

life.augmented

Links

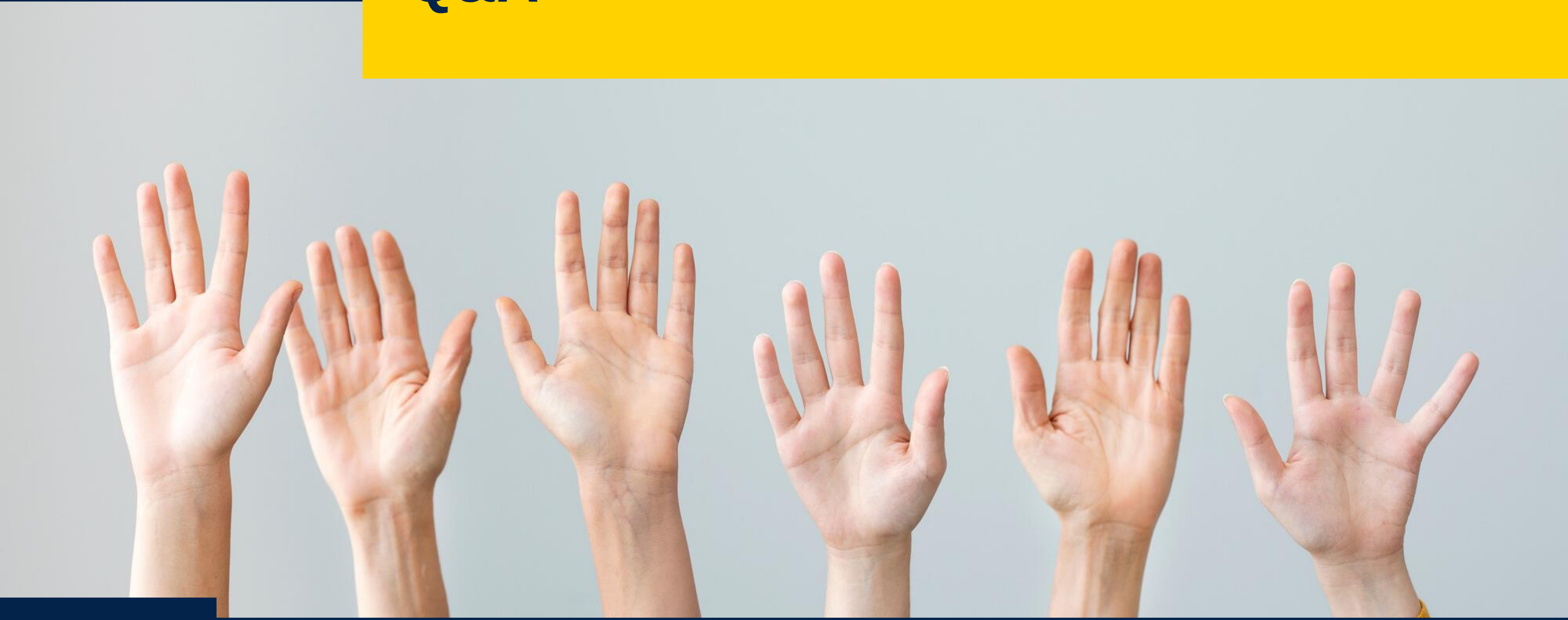
life.augmented

Useful links

- STM32 Graphics Offer: www.st.com/stm32-gui
- STM32 Developer Zone: www.st.com/stm32-dev-zone
- STM32U5 Product Page: www.st.com/stm32u5
- STM32U5 Online Training: www.st.com/stm32u5-online-training
- STM32Cube Expansion Packages: https://www.st.com/en/embedded-software/stm32cube-expansion-packages.html
    - Browse for X-CUBE-TOUCHGFX!
- TouchGFX Support page: https://support.touchgfx.com
- TouchGFX Community page: https://community.st.com/s/topic/0TO0X0000003iw6WAA/touchgfx
- STM32U5A9J-DK page: https://www.st.com/en/evaluation-tools/stm32u5a9j-dk.html

life.augmented

Q&A



life.augmented

Thank you

life.augmented