TouchGFX Dynamic Graphs Demo

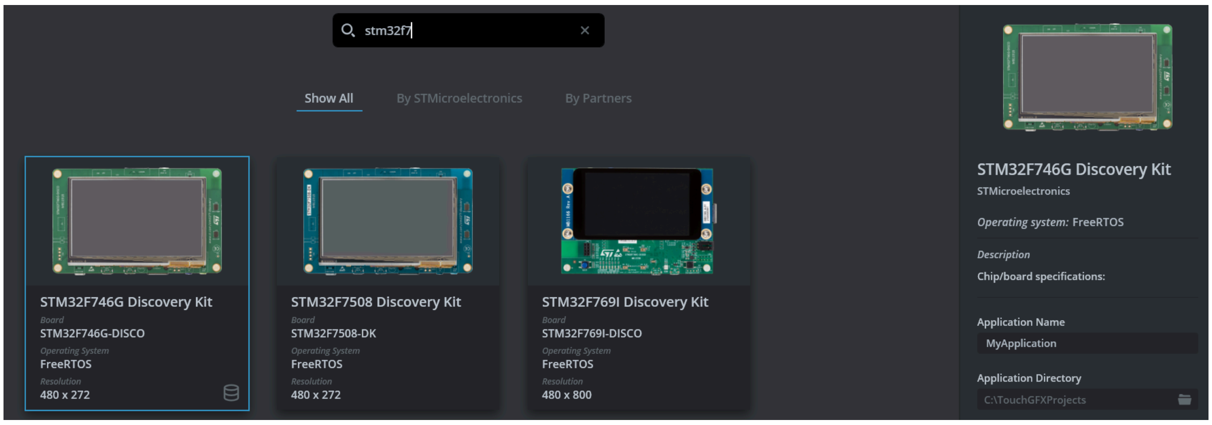
This project demonstrates how to create two dynamic graphs with toggle buttons that control the visibility of graph elements (lines and dots) using TouchGFX Designer and STM32CubeIDE.

In this demo, random values between -20 and 100 are generated to simulate sensor readings.

Project Setup

1. Create a New Application

- Launch **TouchGFX Designer**.
- Select your **target board**.
- Click **Create** to start a new project.



STM32F746G Discovery Kit
Board
STM32F746G-DISCO
Operating System
FreeRTOS
Resolution
480 x 272

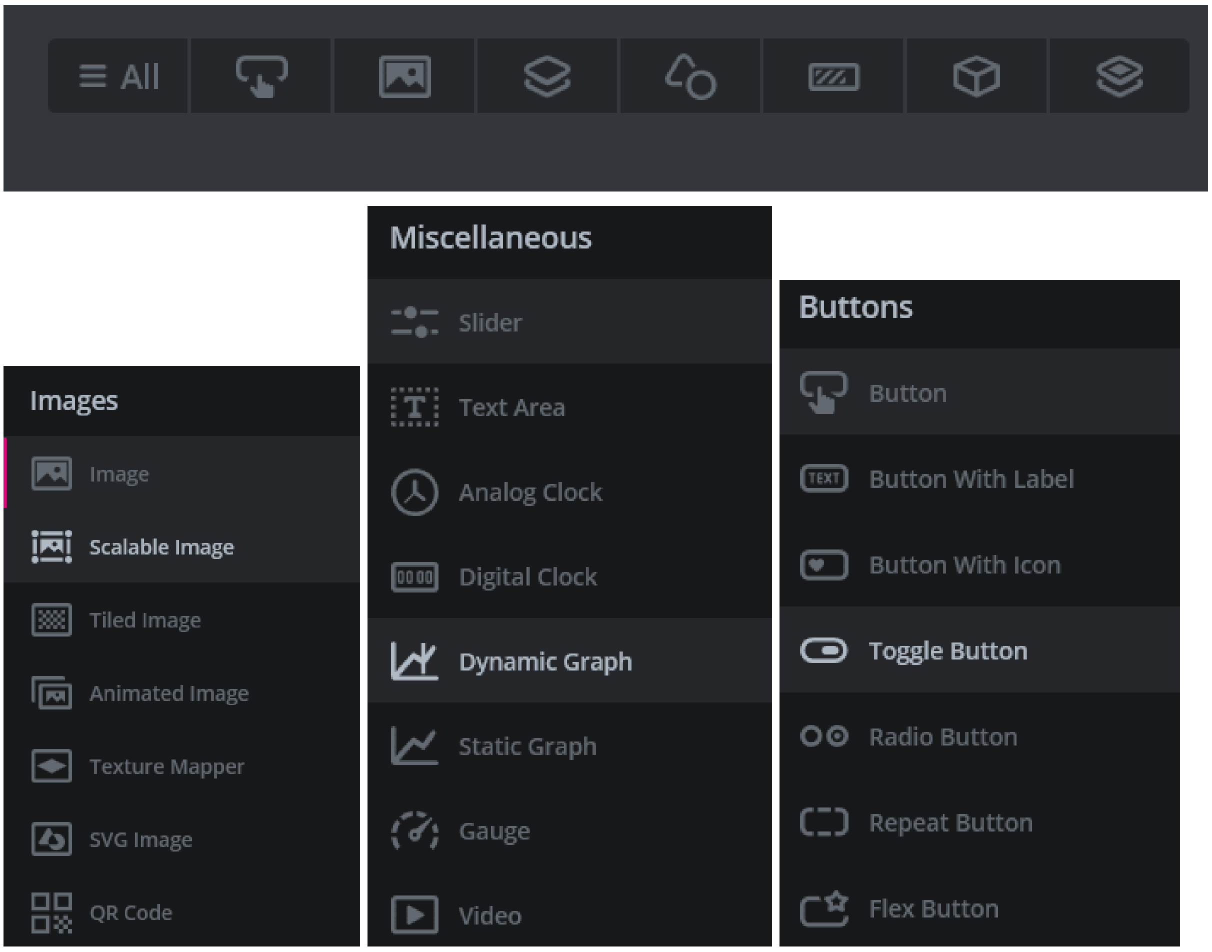
STM32F7508 Discovery Kit
Board
STM32F7508-DK
Operating System
FreeRTOS
Resolution
480 x 272

STM32F769I Discovery Kit
Board
STM32F769I-DISCO
Operating System
FreeRTOS
Resolution
480 x 800

STM32F746G Discovery Kit
STMicroelectronics
Operating system: FreeRTOS
Description
Chip/board specifications:
Application Name
MyApplication
Application Directory
C:\TouchGFXProjects

2. Add UI Elements

- Add a **scalable image** from your file system or the GFX stock.
- Add **two dynamic graphs** (`dynamicGraphBlue`, `dynamicGraphGreen`).
- Add **four toggle buttons** to control visibility:
- Blue Line
- Blue Dots
- Green Line
- Green Dots

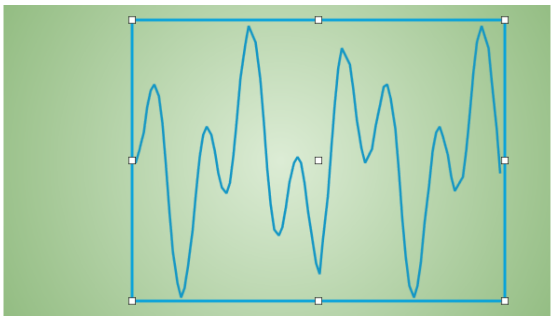


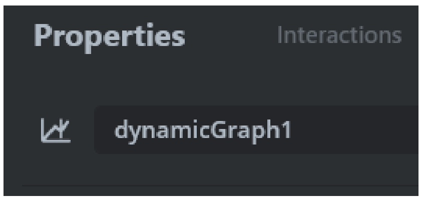
Configure Dynamic Graphs

Graph Properties

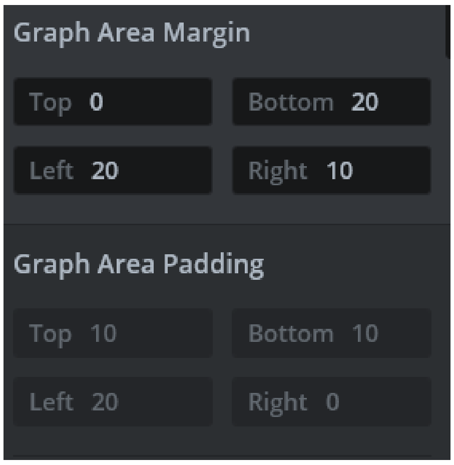
For each graph (e.g., `dynamicGraphBlue`):

- **Graph Area Margins & Padding**: Adjust layout spacing.
- **Dynamic Behavior**: Set to `Scroll` to continuously update X-axis values as new data arrives.
- **Number of Data Points**: Defines how many points are visible on the X-axis at a time.
- **Value Range**: Set Y-axis range to `-20 to 100` (constant).
- **Elements**: Add both `Line` and `Dots`.
- **Grid Lines**: Enable horizontal grid lines.
- **Axis Labels**: Add labels for both X and Y axes.




Properties   Interactions

dynamicGraph1


Graph Area Margin

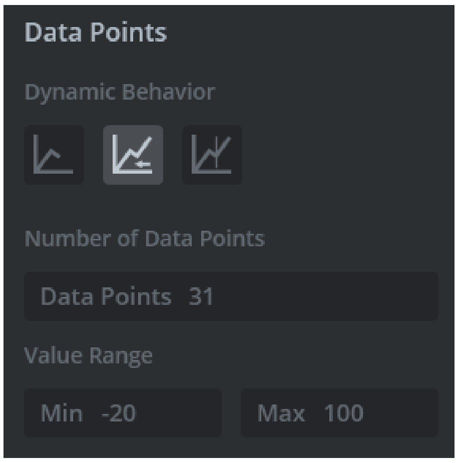
Top 0   Bottom 20

Left 20   Right 10

Graph Area Padding

Top 10   Bottom 10

Left 20   Right 0


Data Points

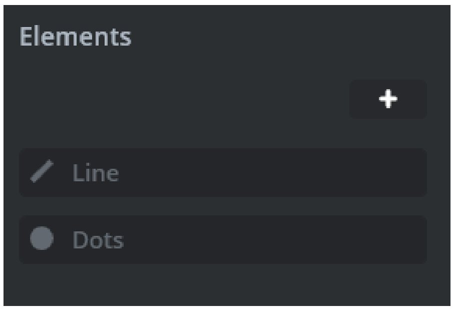
Dynamic Behavior

Number of Data Points

Data Points 31

Value Range

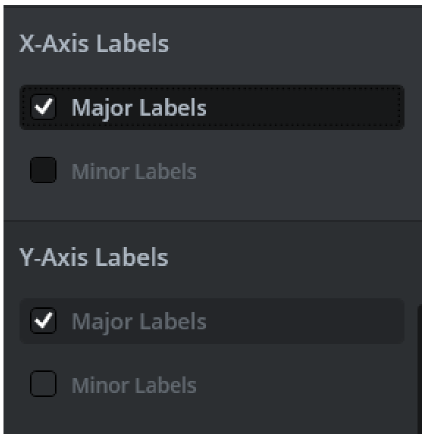
Min -20   Max 100


Elements

+

Line

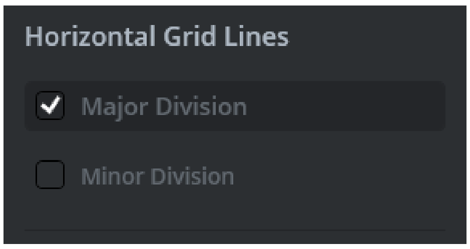
Dots


X-Axis Labels

- [x] Major Labels
- [ ] Minor Labels

Y-Axis Labels

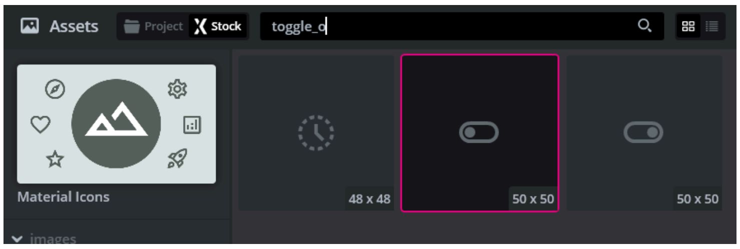
- [x] Major Labels
- [ ] Minor Labels

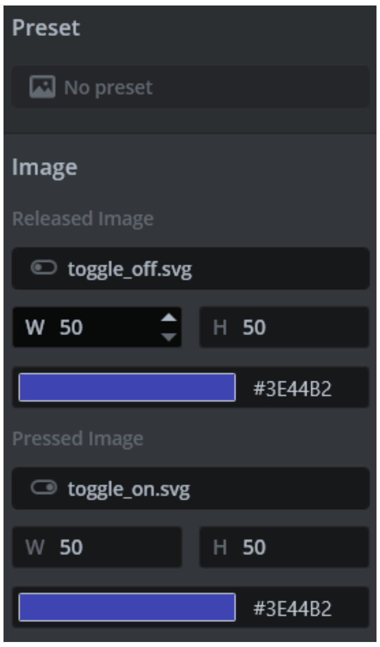

Horizontal Grid Lines

- [x] Major Division
- [ ] Minor Division

Configure Toggle Buttons

- Use stock images: `toggle_off` and `toggle_on`.
- Assign each toggle to control visibility of either line or dots for each graph.





Preset

No preset

Image

Released Image

toggle_off.svg

W 50 H 50

#3E44B2

Pressed Image

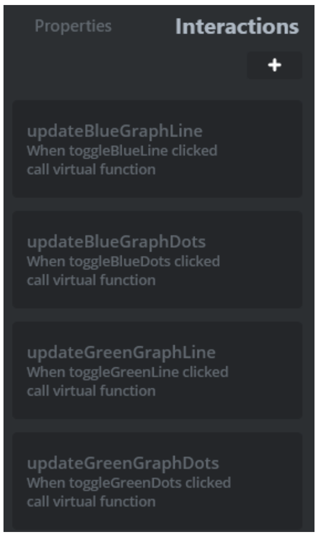
toggle_on.svg

W 50 H 50

#3E44B2

Add Interactions

- Create four interactions, one for each toggle button.
- On button click, call a virtual function:
- `updateBlueGraph()` for blue graph toggles.
- `updateGreenGraph()` for green graph toggles.



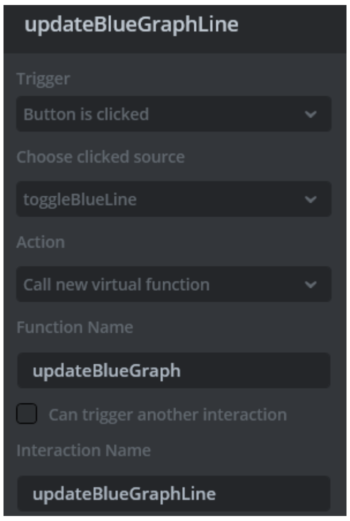
Properties Interactions

updateBlueGraphLine
When toggleBlueLine clicked
call virtual function

updateBlueGraphDots
When toggleBlueDots clicked
call virtual function

updateGreenGraphLine
When toggleGreenLine clicked
call virtual function

updateGreenGraphDots
When toggleGreenDots clicked
call virtual function



updateBlueGraphLine

Trigger: Button is clicked

Choose clicked source: toggleBlueLine

Action: Call new virtual function

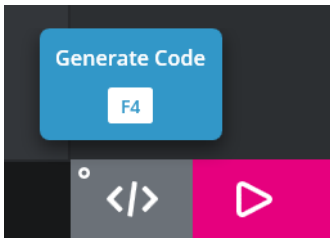
Function Name: updateBlueGraph

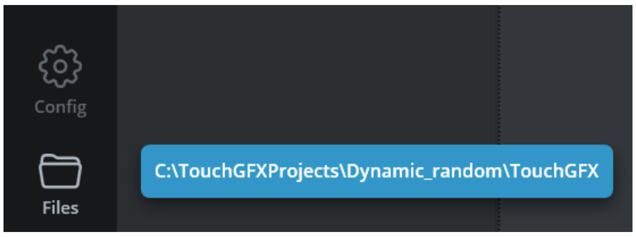
☐ Can trigger another interaction

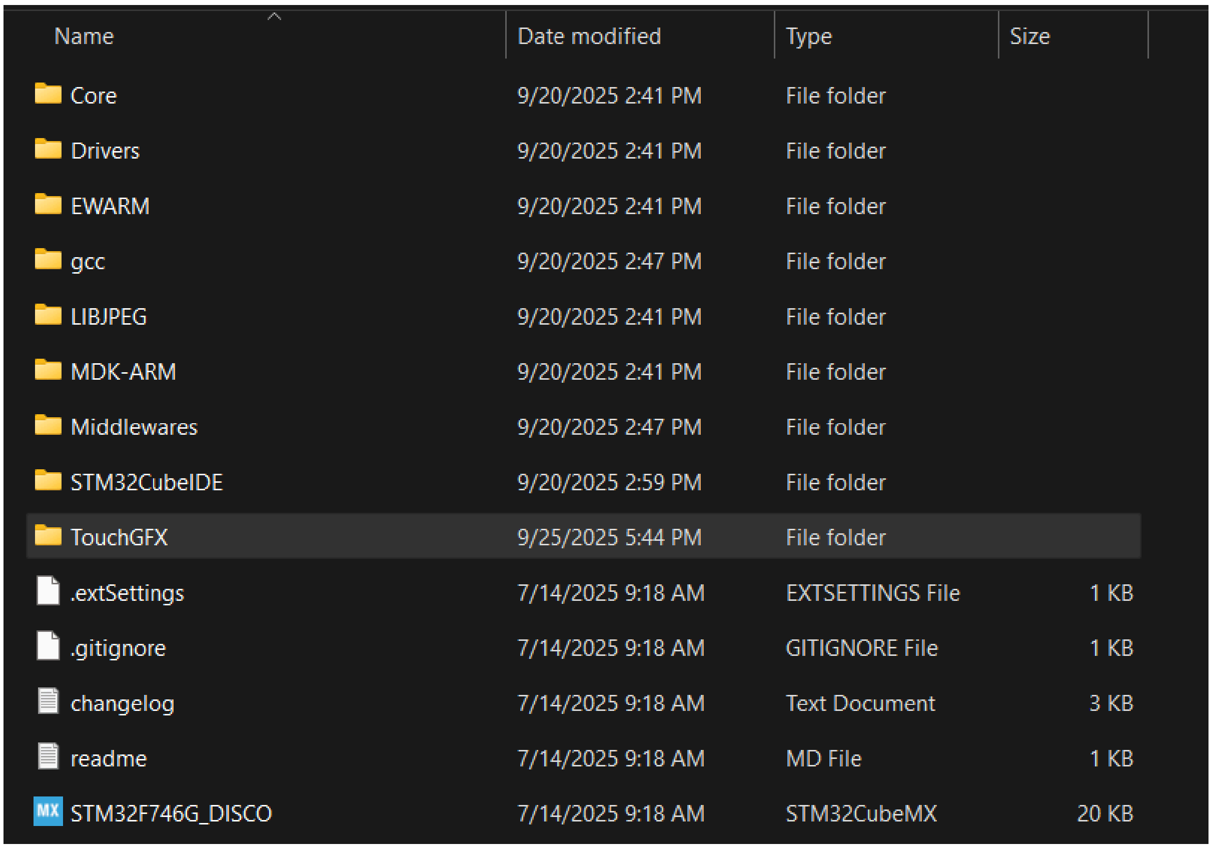
Interaction Name: updateBlueGraphLine

Save and Generate Code

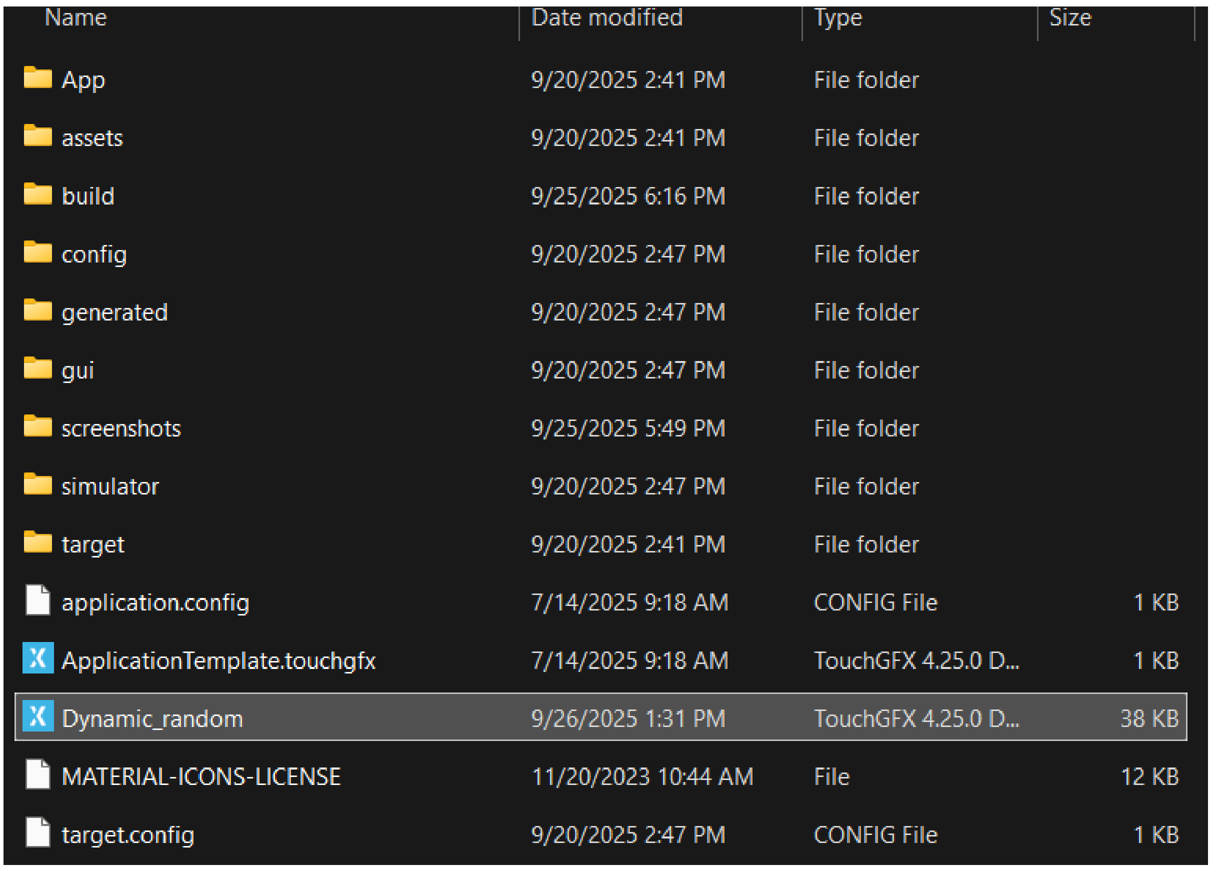
- Click **Save** and **Generate Code**.
- Open the project in STM32CubeIDE: C:\TouchGFXProjects\Dynamic_random\STM32CubeIDE\.cproject







Name	Date modified	Type	Size
Core	9/20/2025 2:41 PM	File folder	
Drivers	9/20/2025 2:41 PM	File folder	
EWARM	9/20/2025 2:41 PM	File folder	
gcc	9/20/2025 2:47 PM	File folder	
LIBJPEG	9/20/2025 2:41 PM	File folder	
MDK-ARM	9/20/2025 2:41 PM	File folder	
Middlewares	9/20/2025 2:47 PM	File folder	
STM32CubeIDE	9/20/2025 2:59 PM	File folder	
TouchGFX	9/25/2025 5:44 PM	File folder	
.extSettings	7/14/2025 9:18 AM	EXTSETTINGS File	1 KB
.gitignore	7/14/2025 9:18 AM	GITIGNORE File	1 KB
changelog	7/14/2025 9:18 AM	Text Document	3 KB
readme	7/14/2025 9:18 AM	MD File	1 KB
STM32F746G_DISCO	7/14/2025 9:18 AM	STM32CubeMX	20 KB

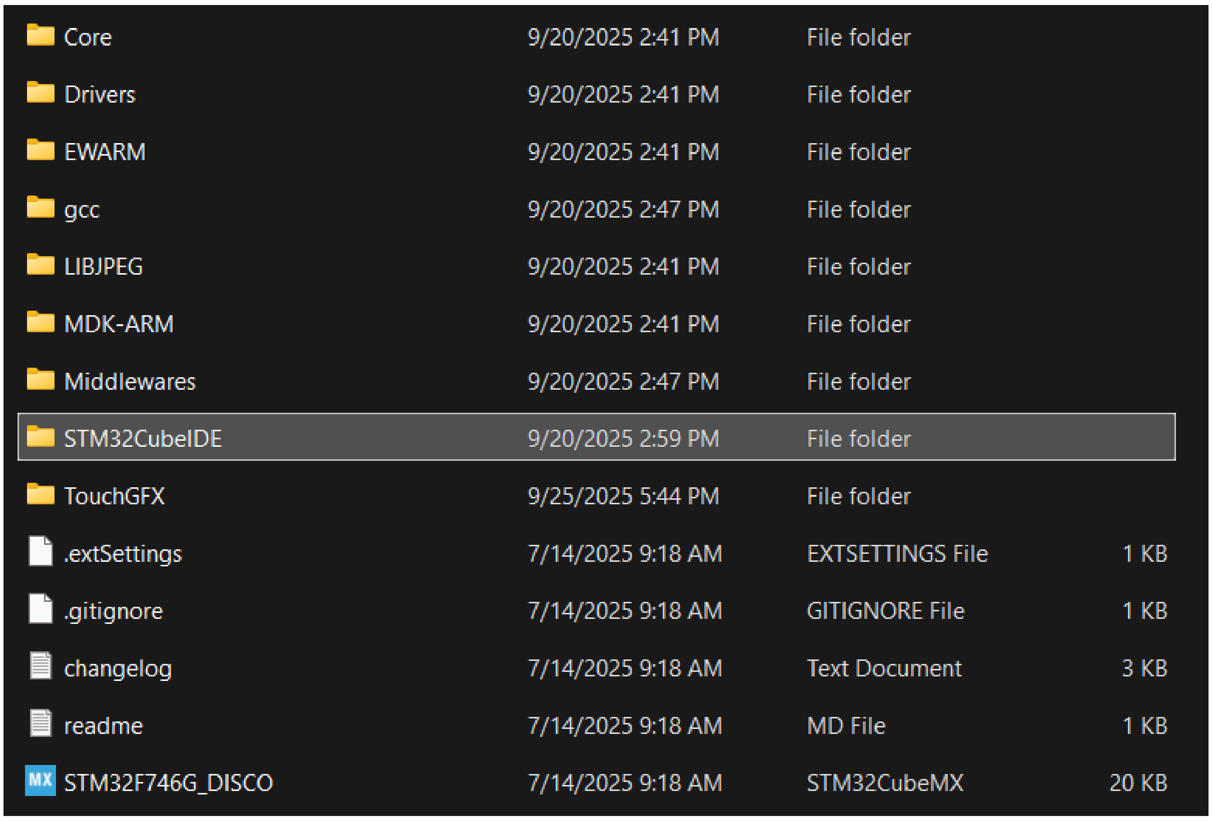


Name	Date modified	Type	Size
App	9/20/2025 2:41 PM	File folder	
assets	9/20/2025 2:41 PM	File folder	
build	9/25/2025 6:16 PM	File folder	
config	9/20/2025 2:47 PM	File folder	
generated	9/20/2025 2:47 PM	File folder	
gui	9/20/2025 2:47 PM	File folder	
screenshots	9/25/2025 5:49 PM	File folder	
simulator	9/20/2025 2:47 PM	File folder	
target	9/20/2025 2:41 PM	File folder	
application.config	7/14/2025 9:18 AM	CONFIG File	1 KB
ApplicationTemplate.touchgfx	7/14/2025 9:18 AM	TouchGFX 4.25.0 D...	1 KB
Dynamic_random	9/26/2025 1:31 PM	TouchGFX 4.25.0 D...	38 KB
MATERIAL-ICONS-LICENSE	11/20/2023 10:44 AM	File	12 KB
target.config	9/20/2025 2:47 PM	CONFIG File	1 KB

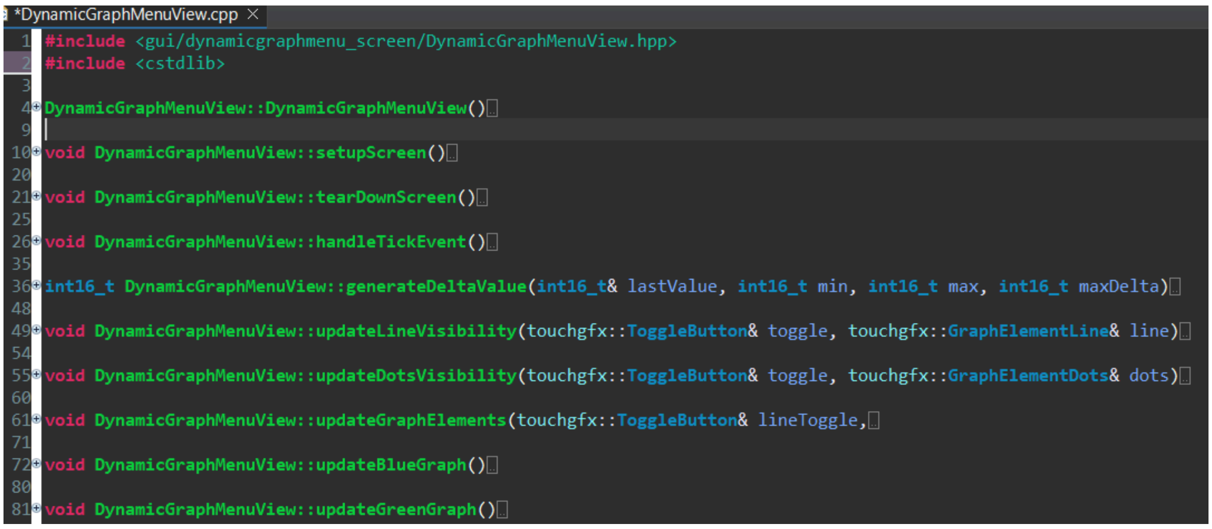
"C:\TouchGFXProjects\Dynamic_random\TouchGFX\generated\gui_generated\src\dynamicgraphmenu_screen\DynamicGraphMenuViewBase.cpp"
"C:\TouchGFXProjects\Dynamic_random\TouchGFX\generated\gui_generated\include\gui_generated\dynamicgraphmenu_screen\DynamicGraphMenuViewBase.hpp"

"C:\TouchGFXProjects\Dynamic_random\TouchGFX\gui\src\dynamicgraphmenu_screen\DynamicGraphMenuView.cpp"
"C:\TouchGFXProjects\Dynamic_random\TouchGFX\gui\include\gui\dynamicgraphmenu_screen\DynamicGraphMenuView.hpp"

"C:\TouchGFXProjects\Dynamic_random\STM32CubeIDE\.cproject"



Core	9/20/2025 2:41 PM	File folder	
Drivers	9/20/2025 2:41 PM	File folder	
EWARM	9/20/2025 2:41 PM	File folder	
gcc	9/20/2025 2:47 PM	File folder	
LIBJPEG	9/20/2025 2:41 PM	File folder	
MDK-ARM	9/20/2025 2:41 PM	File folder	
Middlewares	9/20/2025 2:47 PM	File folder	
STM32CubeIDE	9/20/2025 2:59 PM	File folder	
TouchGFX	9/25/2025 5:44 PM	File folder	
.extSettings	7/14/2025 9:18 AM	EXTSETTINGS File	1 KB
.gitignore	7/14/2025 9:18 AM	GITIGNORE File	1 KB
changelog	7/14/2025 9:18 AM	Text Document	3 KB
readme	7/14/2025 9:18 AM	MD File	1 KB
STM32F746G_DISCO	7/14/2025 9:18 AM	STM32CubeMX	20 KB

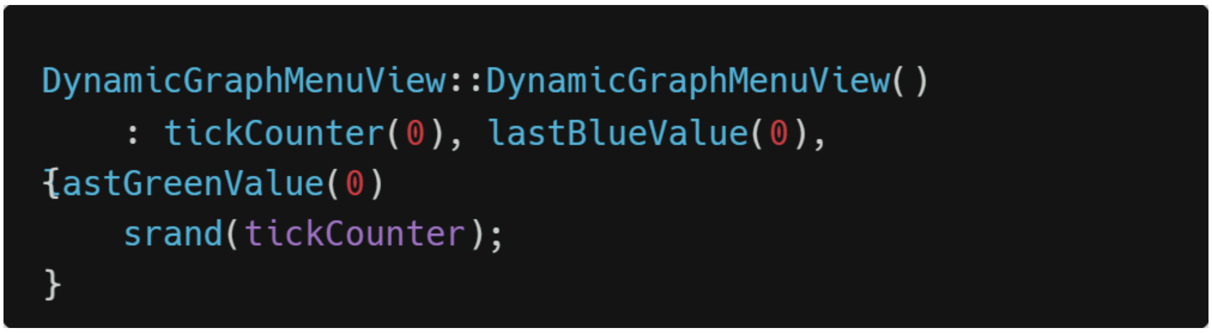


```
#include <gui/dynamicgraphmenu_screen/DynamicGraphMenuView.hpp>
#include <cstdlib>

DynamicGraphMenuView::DynamicGraphMenuView()

void DynamicGraphMenuView::setupScreen()

void DynamicGraphMenuView::tearDownScreen()

void DynamicGraphMenuView::handleTickEvent()

int16_t DynamicGraphMenuView::generateDeltaValue(int16_t& lastValue, int16_t min, int16_t max, int16_t maxDelta)

void DynamicGraphMenuView::updateLineVisibility(touchgfx::ToggleButton& toggle, touchgfx::GraphElementLine& line)

void DynamicGraphMenuView::updateDotsVisibility(touchgfx::ToggleButton& toggle, touchgfx::GraphElementDots& dots)

void DynamicGraphMenuView::updateGraphElements(touchgfx::ToggleButton& lineToggle,

void DynamicGraphMenuView::updateBlueGraph()

void DynamicGraphMenuView::updateGreenGraph()
```

```
#include <gui/dynamicgraphmenu_screen/DynamicGraphMenuView.hpp>
#include <cstdlib>
```

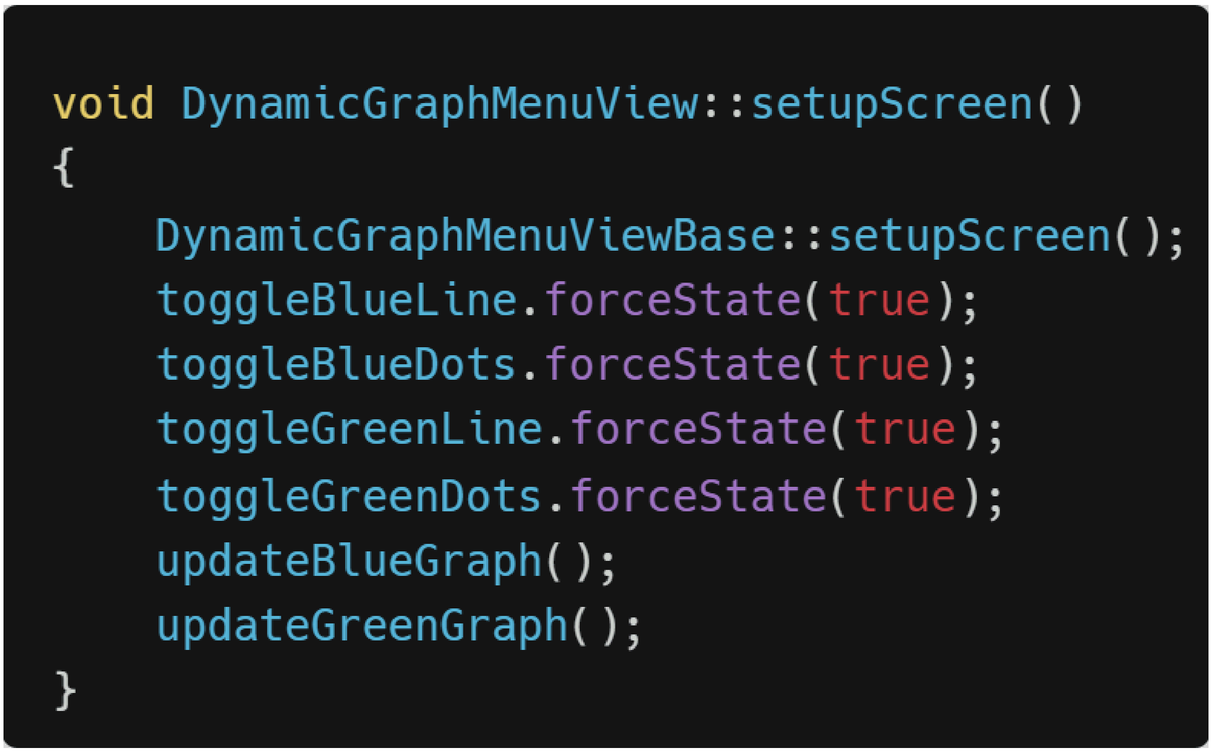
DynamicGraphMenuView Constructor

Creates a new view instance, initializing tickCounter, lastBlueValue, and lastGreenValue to zero. It seeds the random number generator using tickCounter, which will be incremented over time. This setup enables later random value generation for graph simulation and ensures initial values are well-defined for UI logic.



```
DynamicGraphMenuView::DynamicGraphMenuView()
    : tickCounter(0), lastBlueValue(0),
{astGreenValue(0)
    srand(tickCounter);
}
```

setupScreen
Prepares the screen by invoking the base class initialization and programmatically setting toggle buttons to “on” for blue and green lines and dots. The function then calls updateBlueGraph() and updateGreenGraph() to ensure that both graph elements reflect the initial toggle states. This guarantees UI consistency at startup and sets a predictable default visualization.



```
void DynamicGraphMenuView::setupScreen()
{
    DynamicGraphMenuViewBase::setupScreen();
    toggleBlueLine.forceState(true);
    toggleBlueDots.forceState(true);
    toggleGreenLine.forceState(true);
    toggleGreenDots.forceState(true);
    updateBlueGraph();
    updateGreenGraph();
}
```

tearDownScreen
Handles screen teardown by delegating to the base class's teardown logic. This provides a placeholder for future extension if resource cleanup or specific de-initialization is required during navigation away from this screen.

```
void DynamicGraphMenuView::tearDownScreen()
{
    DynamicGraphMenuViewBase::tearDownScreen();
}
```

handleTickEvent
Called on every UI tick, it increments the tick counter and, every tenth tick, generates new simulated data points for both blue and green graphs using generateDeltaValue(). These updates drive live dynamic graph animations and periodic data plotting, synchronizing visual updates with the UI event timeline.

```
void DynamicGraphMenuView::handleTickEvent()
{
    tickCounter++;
    if (tickCounter % 10 == 0)
    {
        dynamicGraphBlue.addDataPoint(generateDeltaValue(lastBlueValue, -20,
100, 5));
        dynamicGraphGreen.addDataPoint(generateDeltaValue(lastGreenValue, -20,
100, 5));
    }
}
```

generateDeltaValue
Generates a random data value based on the last value, within a specified delta, then clamps it between the provided minimum and maximum. Updates the referenced lastValue for subsequent calls. This function enables smooth simulated data variation for demo or testing purposes, ensuring controlled jumps and preventing out-of-range errors.

```
int16_t DynamicGraphMenuView::generateDeltaValue(int16_t& lastValue, int16_t
min, int16_t max, int16_t maxDelta)
{
    int16_t delta = (rand() % (2 * maxDelta + 1)) - maxDelta; // Range: [-
maxDelta, +maxDelta]
    int16_t newValue = lastValue + delta;

    // Clamp to min/max
    if (newValue < min) newValue = min;
    if (newValue > max) newValue = max;

    lastValue = newValue;
    return newValue;
}
```

updateLineVisibility
Sets the visibility (alpha) of a provided line graph element based on the associated toggle button's state, invalidating the line to trigger a redraw. This function modularizes the UI logic for showing or hiding line data, supporting interactive customization of the graph appearance.

```
void DynamicGraphMenuView::updateLineVisibility(touchgfx::ToggleButton&
toggle, touchgfx::GraphElementLine& line)
{
    line.setAlpha(toggle.getState() ? 255 : 0);
    line.invalidate();
}
```

updateDotsVisibility
Similar to updateLineVisibility, but applies to dot elements. Changes dot transparency and requests a redraw in response to toggle state. This ensures dot visibility in the graph mimics user preferences or application mode.

```
void DynamicGraphMenuView::updateDotsVisibility(touchgfx::ToggleButton&
toggle, touchgfx::GraphElementDots& dots)
{
    dots.setAlpha(toggle.getState() ? 255 : 0);
    dots.invalidate();
}
```

updateGraphElements
Synchronizes the visibility of both line and dot elements (and forces a graph refresh) based on the states of their corresponding toggles.

```
void DynamicGraphMenuView::updateGraphElements(touchgfx::ToggleButton& lineToggle,
                                               touchgfx::GraphElementLine& line,
                                               touchgfx::ToggleButton& dotsToggle,
                                               touchgfx::GraphElementDots& dots,
                                               touchgfx::GraphScroll<31>& graph)
{
    updateLineVisibility(lineToggle, line);
    updateDotsVisibility(dotsToggle, dots);
    graph.invalidate();
}
```

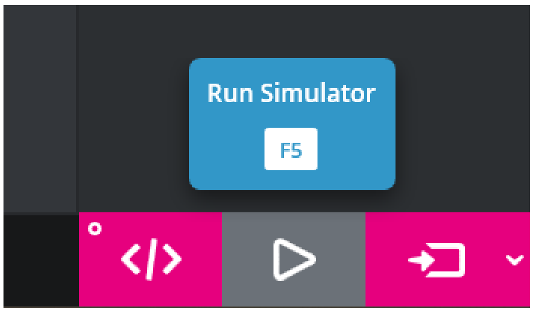
updateBlueGraph
Calls updateGraphElements for the blue graph, updating line and dot appearance and then invalidating the entire blue graph container to force a complete redraw. This is triggered during startup, or whenever blue-related toggles change.

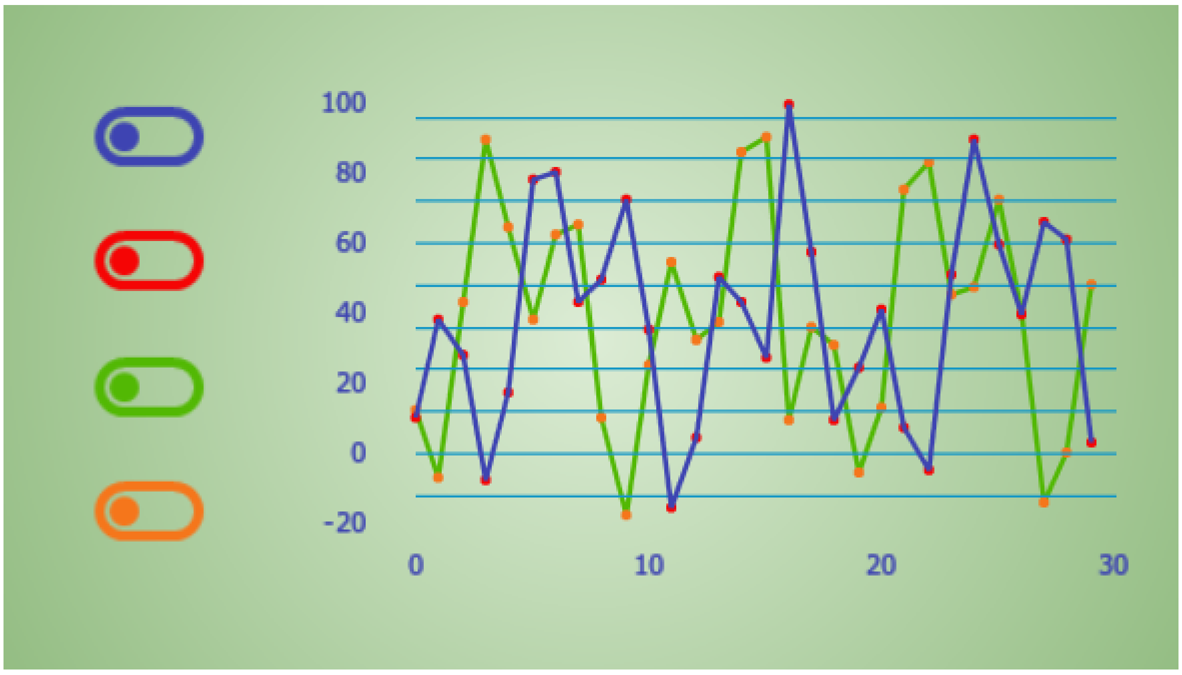
```
void DynamicGraphMenuView::updateBlueGraph()
{
    updateGraphElements(toggleBlueLine, dynamicGraphBlueLine1,
                        toggleBlueDots, dynamicGraphBlueDots1,
                        dynamicGraphBlue);

    dynamicGraphBlue.invalidate();
}
```

updateGreenGraph
Analogous to updateBlueGraph, but for the green data series.

```
void DynamicGraphMenuView::updateGreenGraph()
{
    updateGraphElements(toggleGreenLine, dynamicGraphGreenLine1,
                        toggleGreenDots, dynamicGraphGreenDots1,
                        dynamicGraphGreen);
    dynamicGraphGreen.invalidate();
}
```

Go to Designer and Run Simulator (or flash the code to your board).


Run Simulator
F5


100
80
60
40
20
0
-20
0
10
20
30