TouchGFX
Embedded Graphics
- Basic Concepts

Agenda

1. Introduction
2. Hardware
3. Color Formats
4. Framebuffer
5. Graphics Engine
6. Main Loop
7. Main Loop & Framebuffer
8. Performance
9. Operating Systems

life.augmented

Introduction

Introduction

Goal of this presentation

- General knowledge about embedded graphics
  - Basic hardware
  - Color formats
  - Framebuffer
- Basic knowledge about the TouchGFX Engine
  - The Graphics Engine
  - The Main Loop
  - Performance
  - Operating System

Introduction

Further reading

- You will find a lot of help afterwards in the TouchGFX documentation site: https://support.touchgfx.com/
- Slides in this presentation will refer to relevant documentation pages. Links will be in the lower right-hand corner of the slides
- The presentation is based on the TouchGFX documentation: https://support.touchgfx.com/docs/basic-concepts/embedded-graphics

Introduction

TouchGFX Development

- Main Components:

- Main Activities

- In this presentation, we will not focus on a specific component or activity as it is an introduction to the basic elements. We will instead focus on important know-how elements when working with components and activities related to TouchGFX Development

life.augmented

Documentation Link: Development Introduction

Introduction

- **What is Embedded Graphics?**
  - *What it can be*
    - All embedded devices with a graphical display
    - Modern smartphone, with high resolution and 3D animations
    - Old 8 bit MCU and a character og segment displays
  - *What it is not*
    - A personal computer and tablets
  - *What it is to TouchGFX*
    - UIs on STM32 microcontroller (32 bit MCUs).
    - Interactive applications in 2-2.5D.
    - User interface running at 30-60 frames per second.

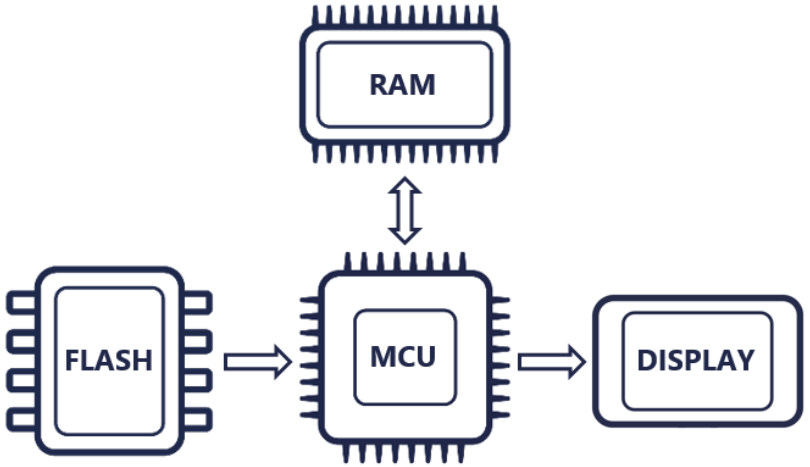
life.augmented

Documentation Link: Embedded Graphics

Hardware

Hardware

- The basic hardware can be divided into 4 parts
  - MCU
    - Heavy Lifting
  - RAM
    - Framebuffer
  - Flash
    - Images, Fonts, Texts
  - Display



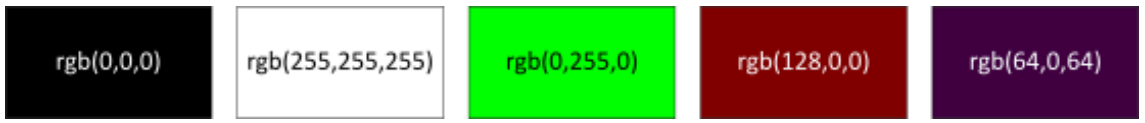
- More information can be found in the documentation:

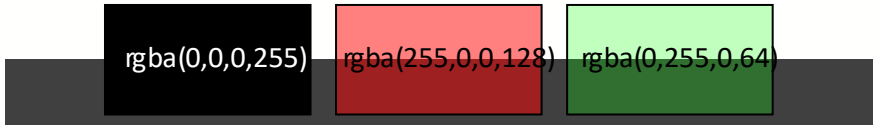
  Hardware Selection

Color Formats

Color Formats

- Digital images can be divided into small single components called "pixels"
- Pixels
  - Single color
  - RGB
  - 0 to 255
  - Opacity (Alpha - RGBA)



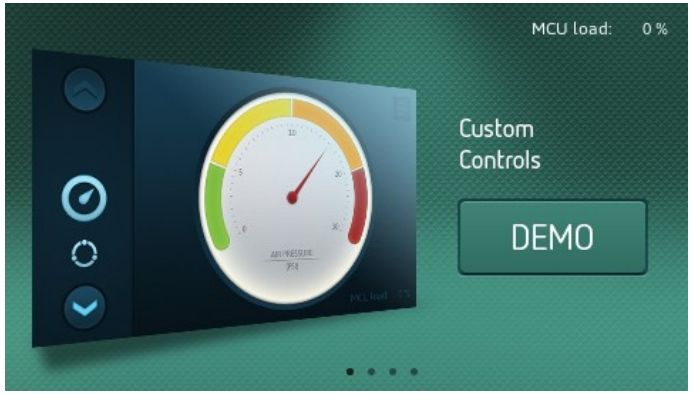


life.augmented

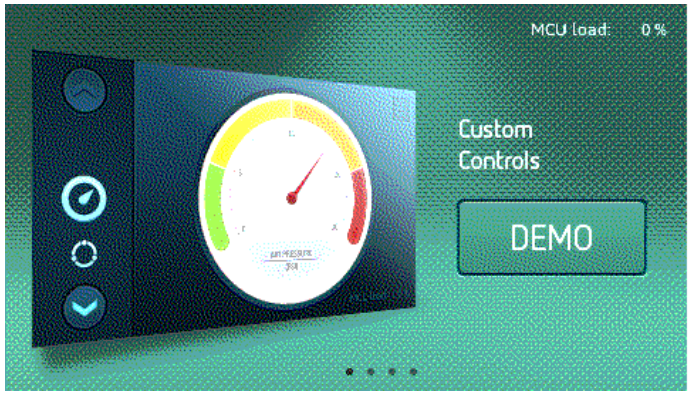
Documentation Link: Color Formats

Color Formats

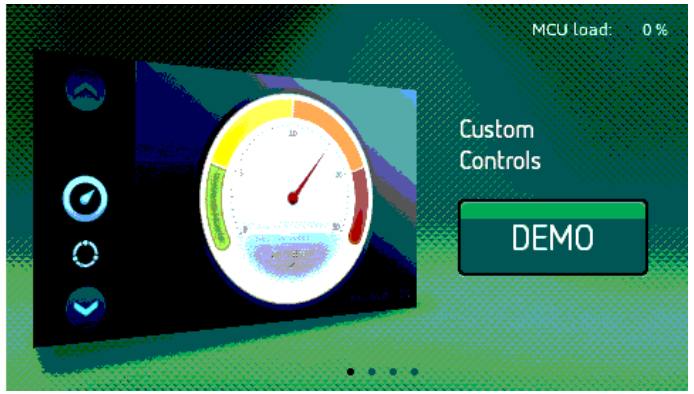
- Color Depths
  - Amount of bits to describe a pixel
    - Bits per pixel (BPP)
    - Range 1 – 32
  - Affects visual quality at the cost of memory
  - The higher BPP,
    the more memory is needed
  - Low BPP can be
    compensated with
    Dithering



24 bpp



8 BPP – with dithering



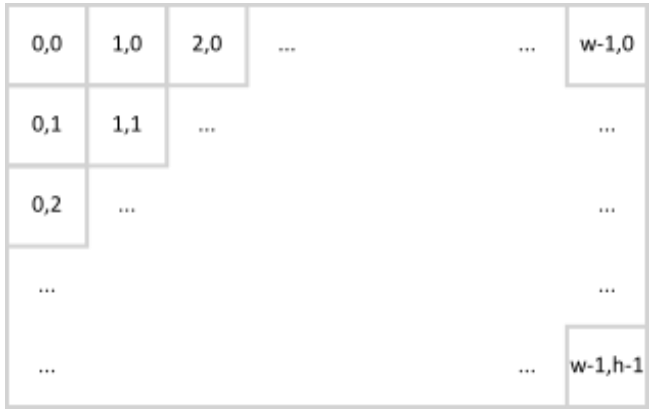
8 BPP – without dithering

Documentation Link: Color Depth

Framebuffer

Framebuffer

- What is a Framebuffer?
  - A framebuffer contains pixel information
  - A 2D memory block with the image to display
  - The framebuffer size is determined by the screen resolution of the display and the pixel BPP



The 2D framebuffer, indexable by x, y

From framebuffer pixel values to colors on a display

Documentation Link: Framebuffer

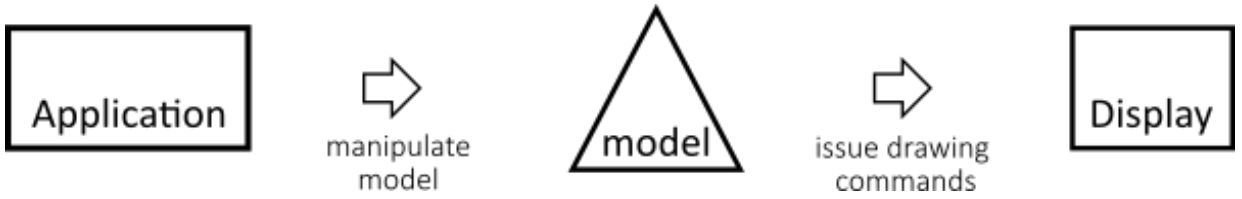
Framebuffer

- The amount of framebuffers can be divided into 3 strategies
  - More than one (usually two)
  - One
  - Less than one
- Strategy impact on performance
  - More than 1 gives better performance
  - But cost more memory
- Calculating framebuffer cost
  - The cost is $width \times height \times bpp \times nr.of\ frambuffers \div 8\ bits$
  - Screen resolution at 800 x 480, color depth of 24 BPP and 2 framebuffers
  - $800 * 480 * 24\ bpp * 2\ framebuffers \div 8\ bits = 2.304\ Mb$

Graphics Engine

Graphics Engine

- TouchGFX is a retained mode graphics engine
  - User manipulates an abstract model
  - The engine translates the model to drawing operations



- The components in the model is referred to as "widgets"
  - Graphics displayed by interacting with the widgets
  - Interaction is done in the Designer or C++ code
  - Handling of the drawing

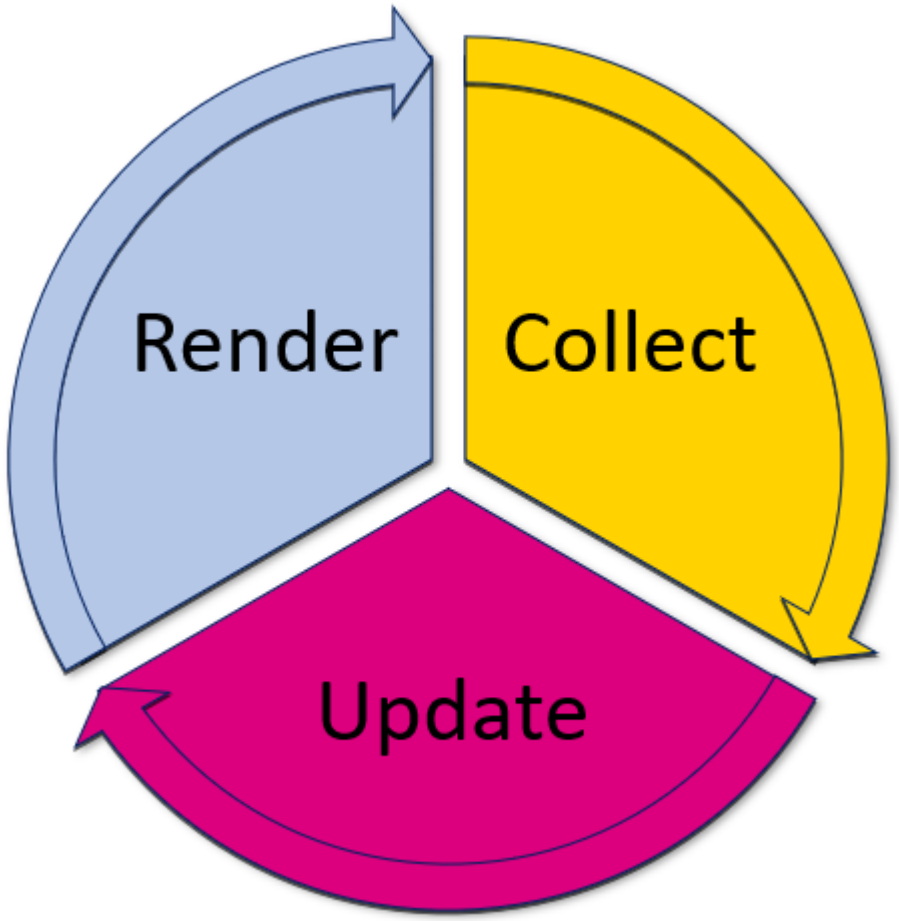
life.augmented

Documentation Link: Graphics Engine

Main Loop

Main Loop

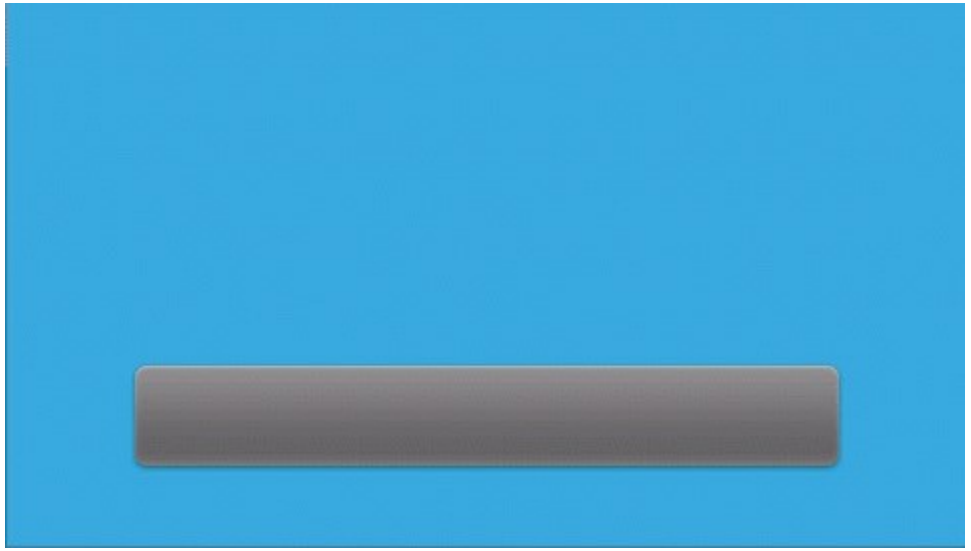
- TouchGFX runs as an “infinite” loop
  1. Collecting events
     - Touchscreen and buttons
     - Backend (Ticks)
  2. Updating
     - Changes the “state” of the widgets based on the events collected
  3. Rendering
     - TouchGFX draws relevant updates in the framebuffer
  - After the loop, TouchGFX waits for a signal and runs the loop again



Documentation Link: Main Loop

Main Loop

- Example

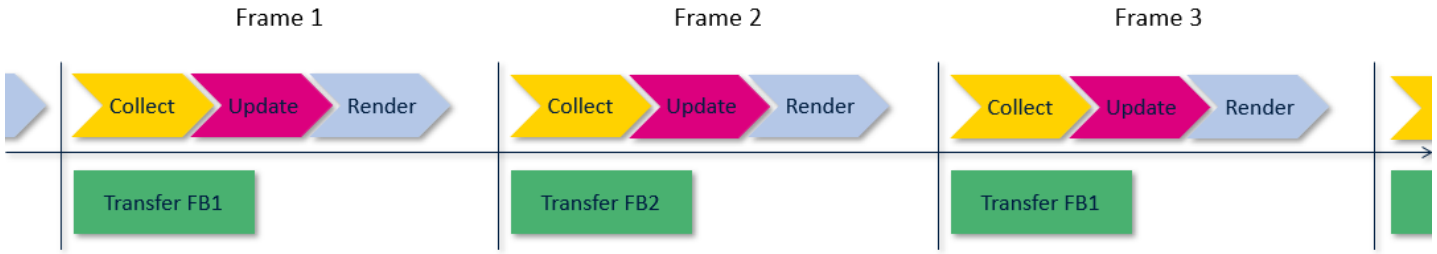


Documentation Link: Main Loop

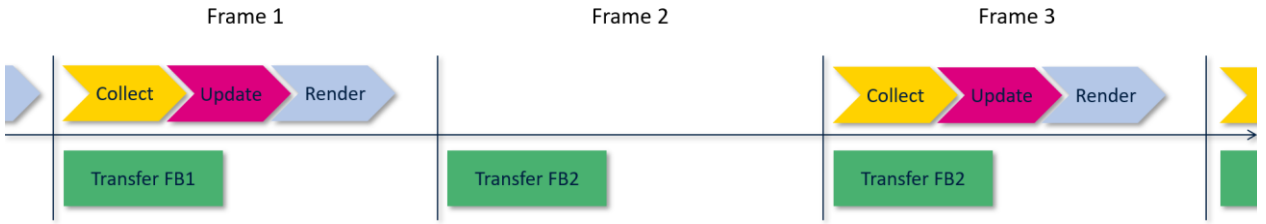
Framebuffer & Main Loop

Framebuffer & Main Loop

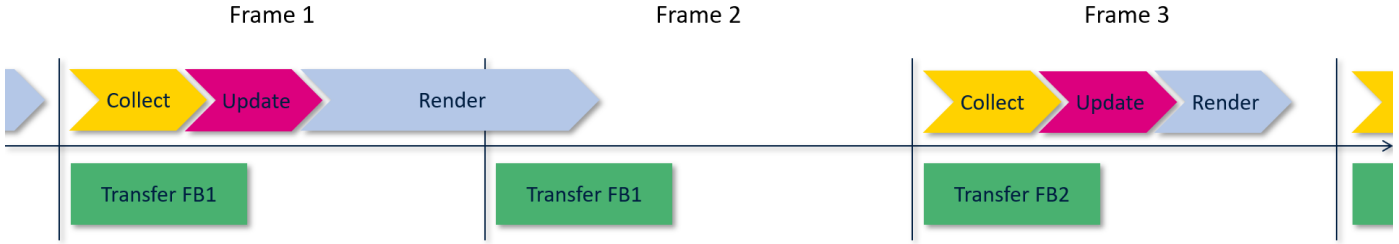
- Effect of the framebuffer strategy
  - More than one framebuffer
    - When a loop is done, the framebuffer is switched



    - Nothing has changed and the framebuffer is retransmitted



    - If the rendering is to slow, the the old framebuffer is retransmitted

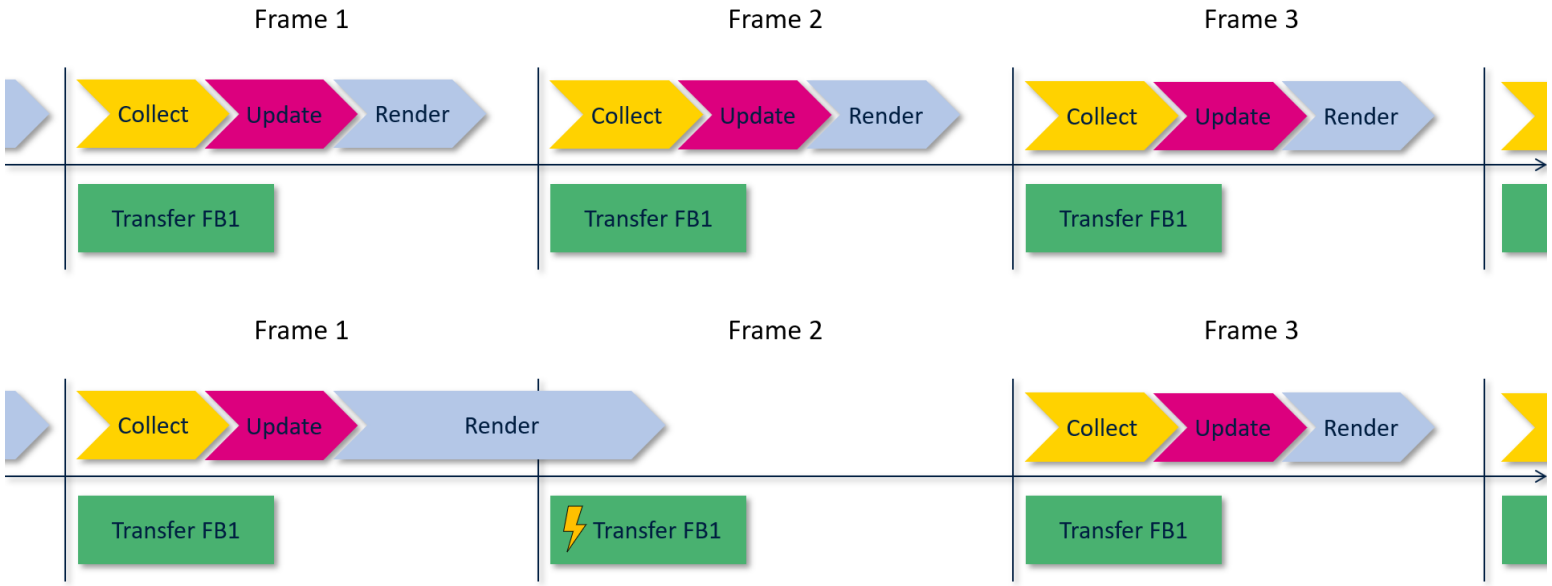


Documentation Link: Handling the framebuffers

life.augmented

Framebuffer & Main Loop

- One framebuffer
  - We have to draw in the same framebuffer that is being transmitted
  - Creates risk for transmitting part of the old frame “tearing”
  - Solutions are only drawn when transfer is done or in the already transferred part of the framebuffer



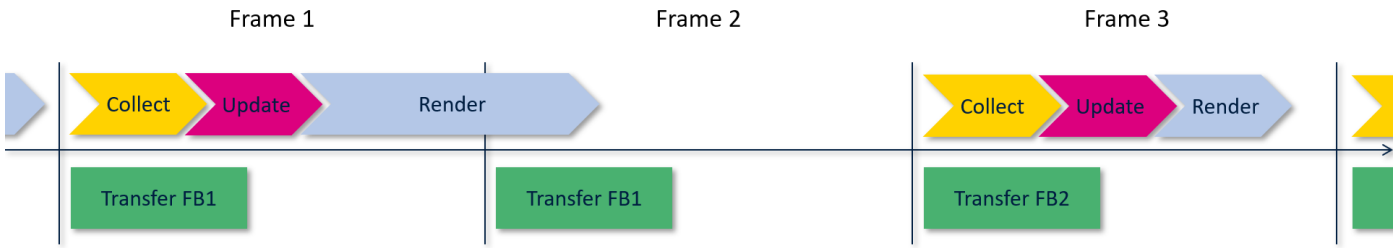
Documentation Link: Handling the framebuffers

life.augmented

Performance

Performance

- Good performance
  - Desired graphics and animations
  - High frame rate
- Usually the frame rate is around 60 Hz
  - A loop therefor has 1 s / 60 = 0.01667 s = 16.67 ms
- Using more time results in a lower frame rate

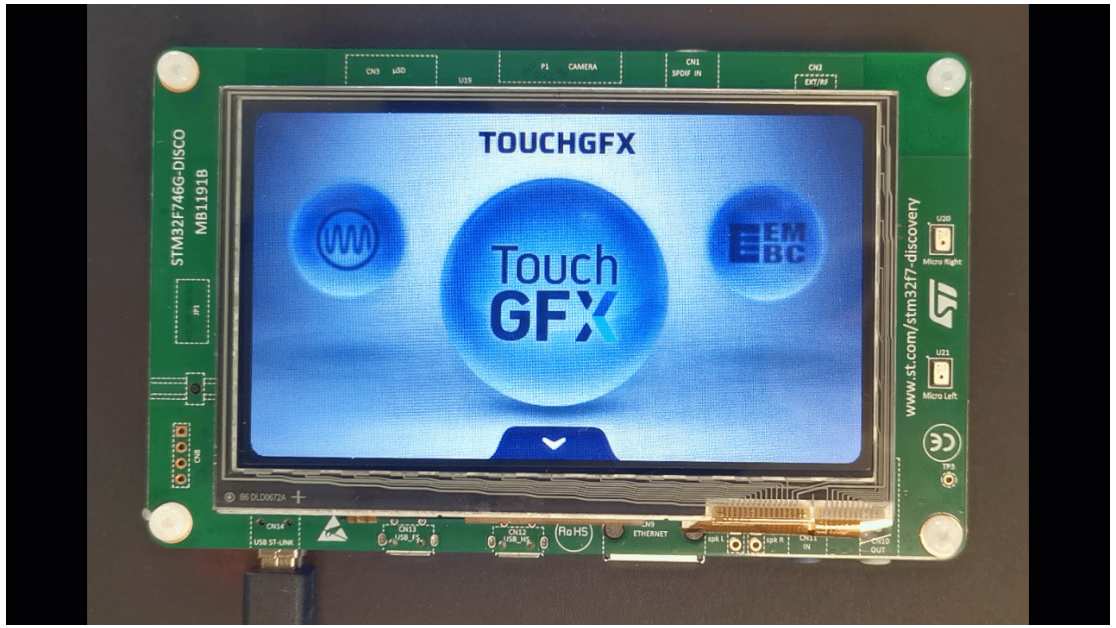


life.augmented

Documentation Link: Performance

Performance

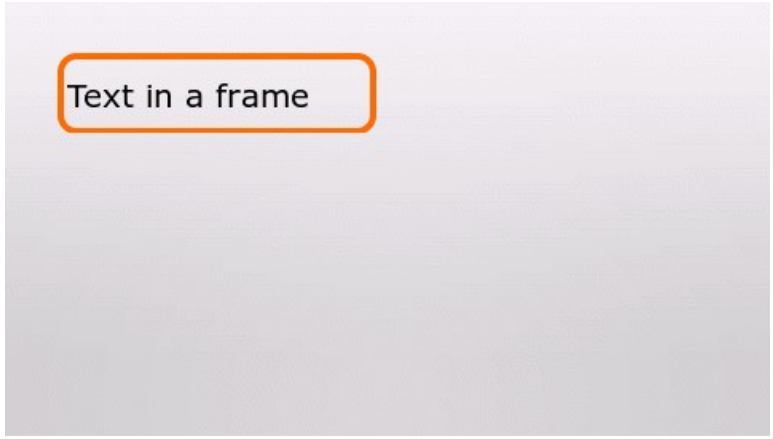
- Example



Documentation Link: Performance

Performance

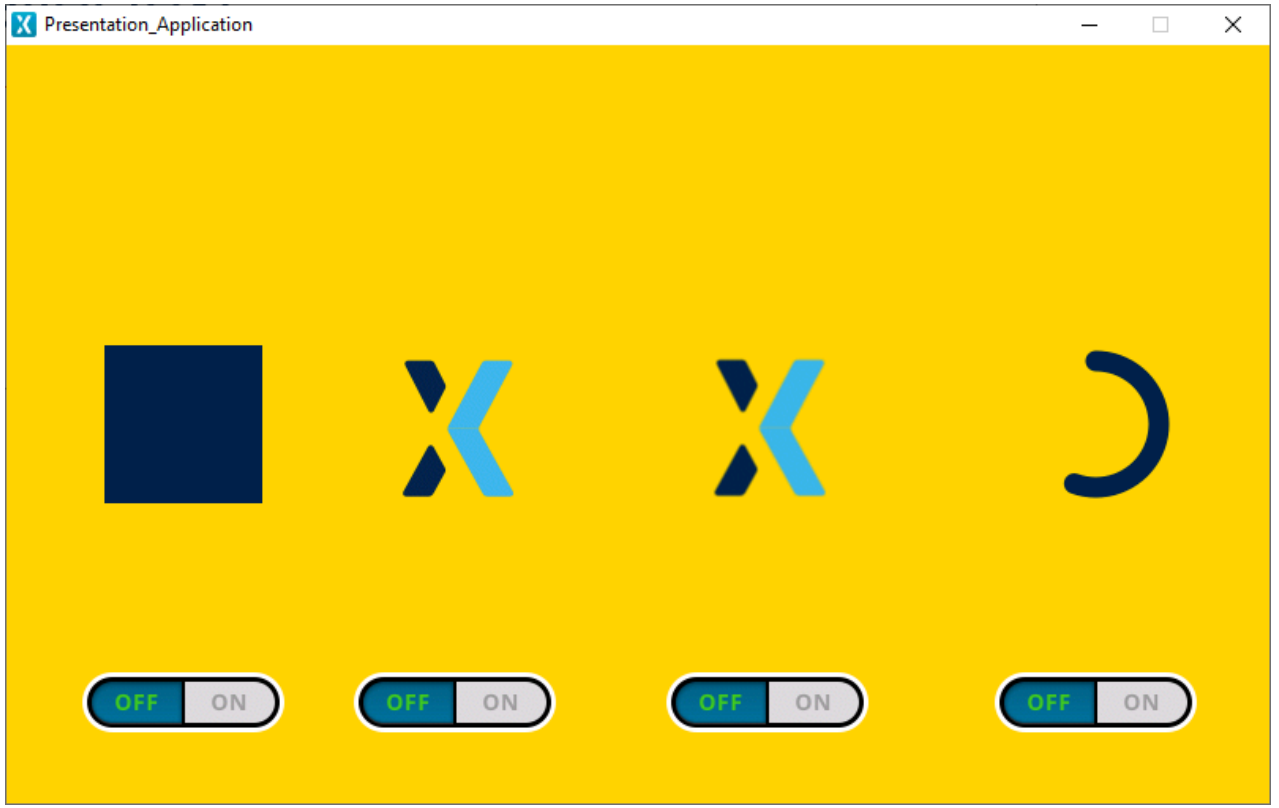
- What affects the rendering time?
  - The amount of the updated screen
    - Larger area = more computation
  - Layers in graphics
    - Each layer requires rendering
  - Complexity of the rendered pixel
    - Some widgets are heavier to render than others
    - Transparency adds to complexity
  - Hardware support for rendering
    - Offloads the MCU
    - Chrom-ART



Documentation Link: What Affects the Rendering Time?

Performance

- Example



Documentation Link: What Affects the Rendering Time?

Operation Systems

Operation Systems

- Embedded devices handling more than the UI
- RTOS
  - Interleaving tasks
    - Ensure that the UI is not blocked by another proces or the like
  - Communication between tasks
    - Utilizing RTOS message queue
  - FreeRTOS
- No RTOS
  - Small setups with low complexity

More Info

life.augmented

More info

- To learn more about the hardware and what to select, read the Hardware Selection Introduction
- To get started with the UI development, read the UI Development Introduction
- Or watch the presentation or workshop

  UI Development – Fundamentals

  UI Development - Getting Started

life.augmented

Thank you

life.augmented