



arm

初探 Arm-2D

填补空白还是屋上架屋

王卓然

2021-5-21

内容提要

- 什么是Arm-2D
- Arm-2D开发者模型简介
- Arm-2D开发范例
- Q&A

```
git clone https://github.com/ARM-software/EndpointAI
```



arm

Arm-2D是什么

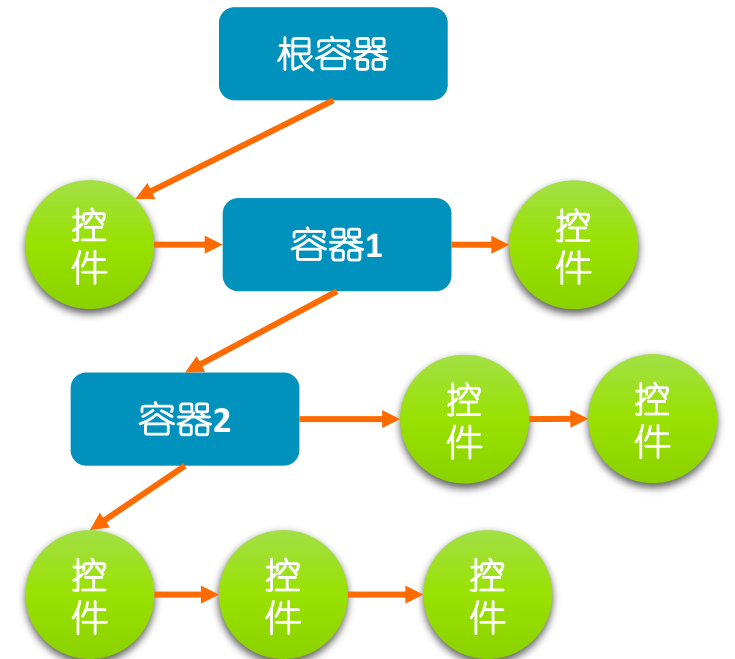
Arm-2D是GUI么？

常见GUI基本构成

- GUI的基本框架（GUI Software Framework）
 - 控件树（Element Tree）
 - 消息处理机制
 - Boxing Mode
 - 资源管理（字体、图片资源等等）
 - 特效和动画支持服务
 - 硬件适配、抽象和加速
- GUI控件库
 - 常用的控件集合
 - 控件的皮肤和式样库
- 用户接口
 - 设计器
 - 对象库、模板库、例程

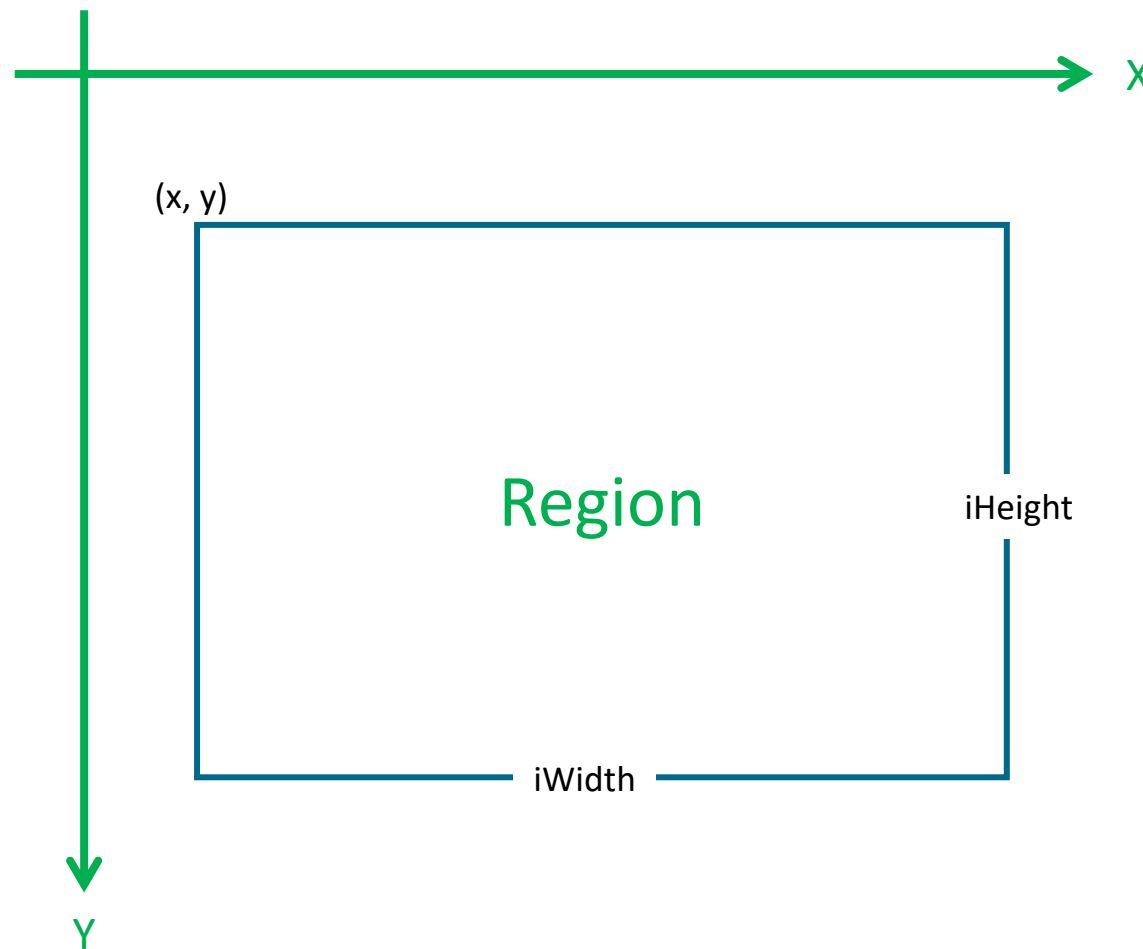
控件树

- 容器（Container）和控件（Control）
 - 控件是控件树的基本单位
 - 容器是一种特殊的控件——它可以容纳其它控件
 - 不同的容器对内部的控件可以有完全不同的组织形式
 - “放任不管”式容器
 - 流式布局（Stream Container）
 - 线性布局（Line Stream Container）
 -
 - Z-Order
- 对同一个控件树，不同的消息有完全不同的遍历策略
 - 界面绘制（Refresh）通常采用先序遍历或者广度优先遍历
 - 触摸事件处理通常采用深度优先搜索



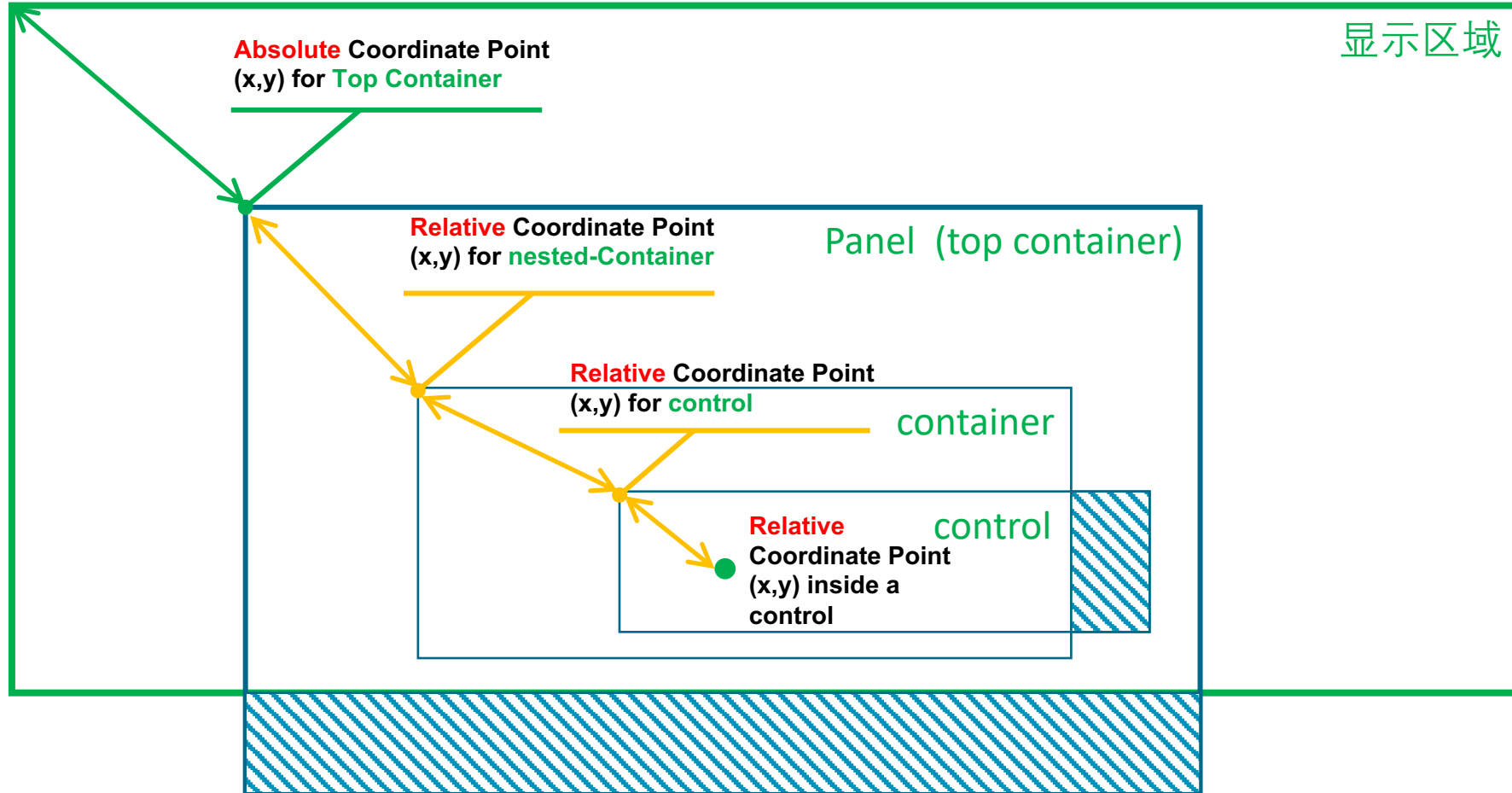
Boxing Mode

- Boxing Mode 是关于界面布局的模型
- Region 是Boxing Mode的基本单位
 - 位置信息 (Location)
 - int16_t
 - 2D环境下使用平面直角坐标系
 - 左上角为 (0,0)
 - Y 轴正方向向下
 - 尺寸信息 (Size)
 - int16_t
 - 宽度 (Width)
 - 高度 (Height)

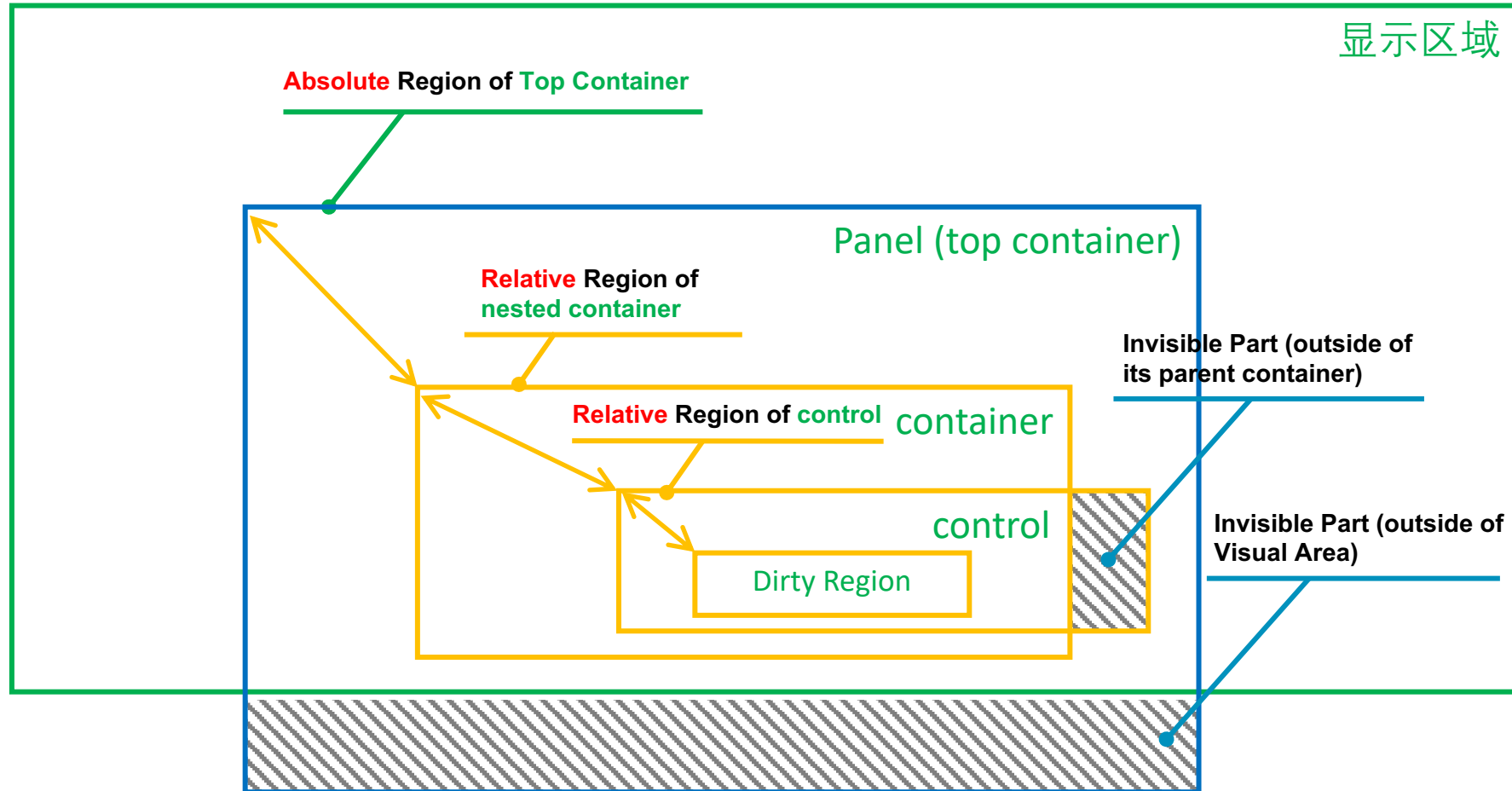


绝对坐标和相对坐标

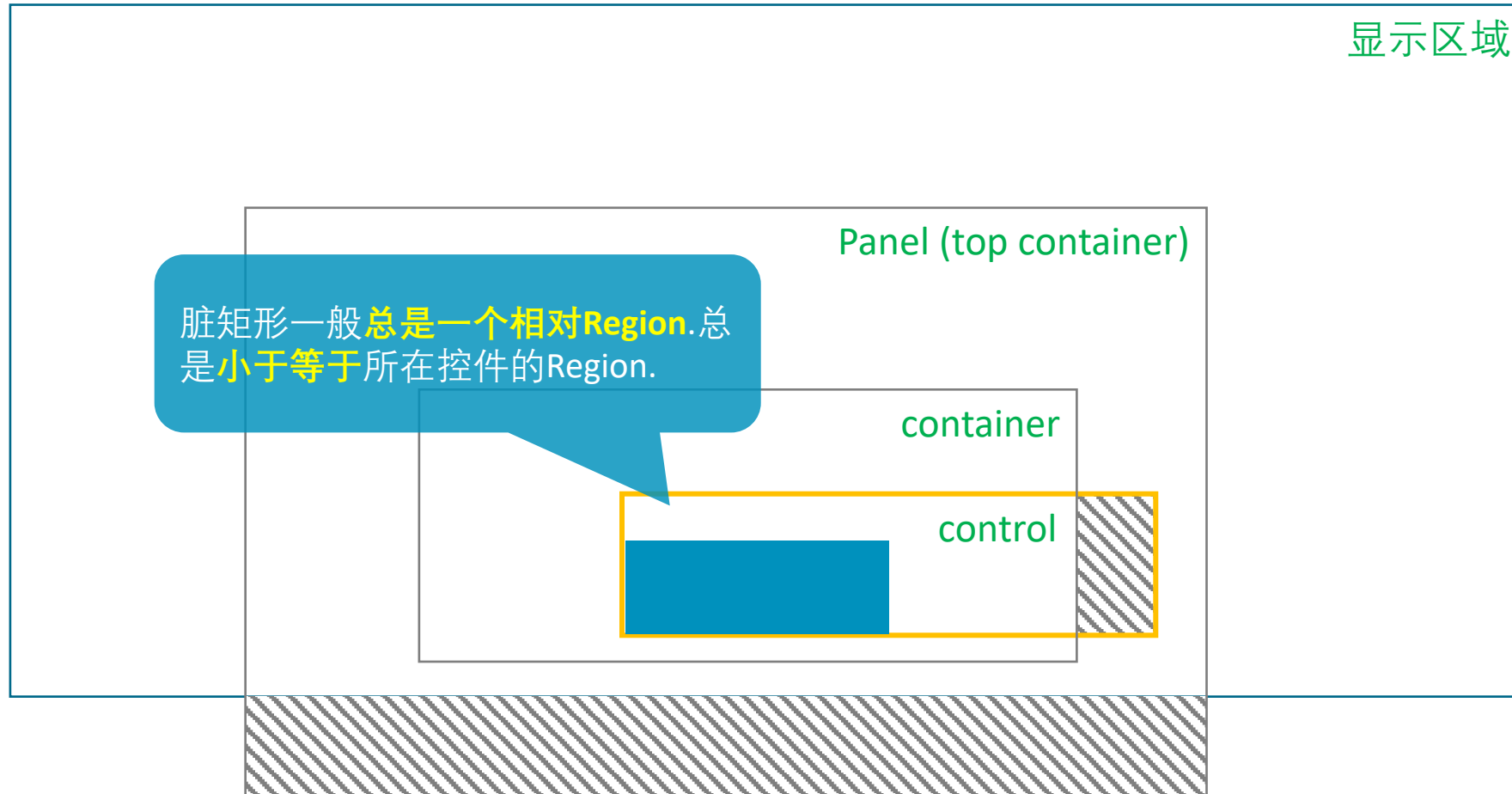
Absolute Base Point (0,0)



相对Region和绝对Region



脏矩形 (Dirty Region)



常见GUI基本构成

- GUI的基本框架（GUI Software Framework）

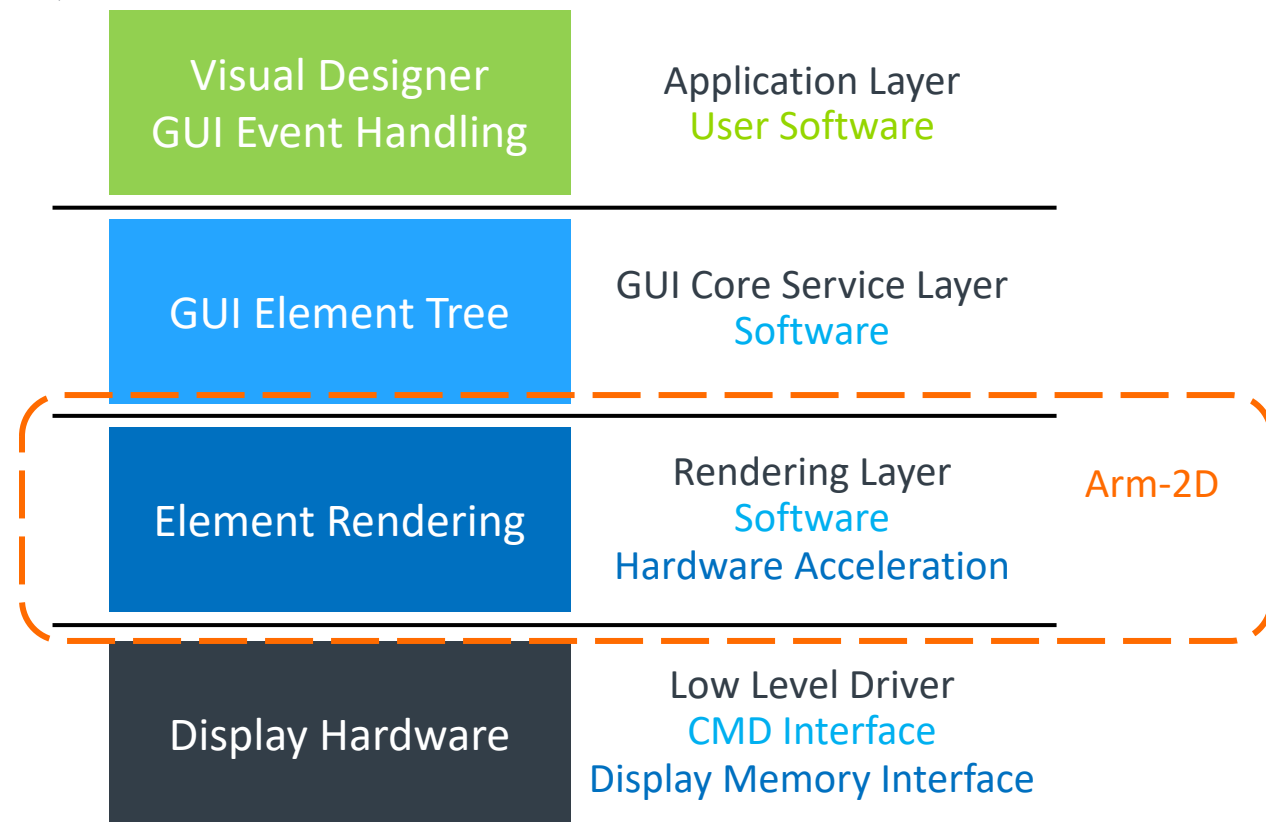
- 控件树（Element Tree）
 - 消息处理机制
 - Boxing Mode
- 资源管理（字体、图片资源等等）
- 绘制、特效和动画支持服务
- 硬件适配、抽象和加速

- GUI控件库

- 常用的控件集合
- 控件的皮肤和式样库

- 用户接口

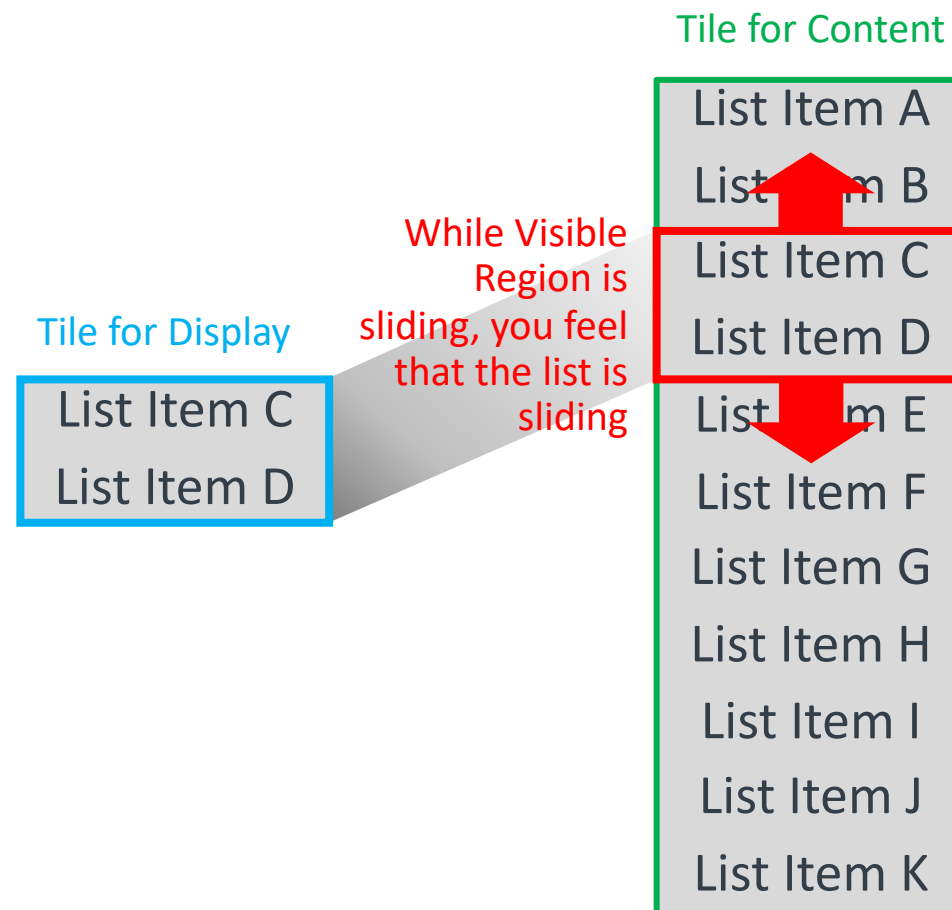
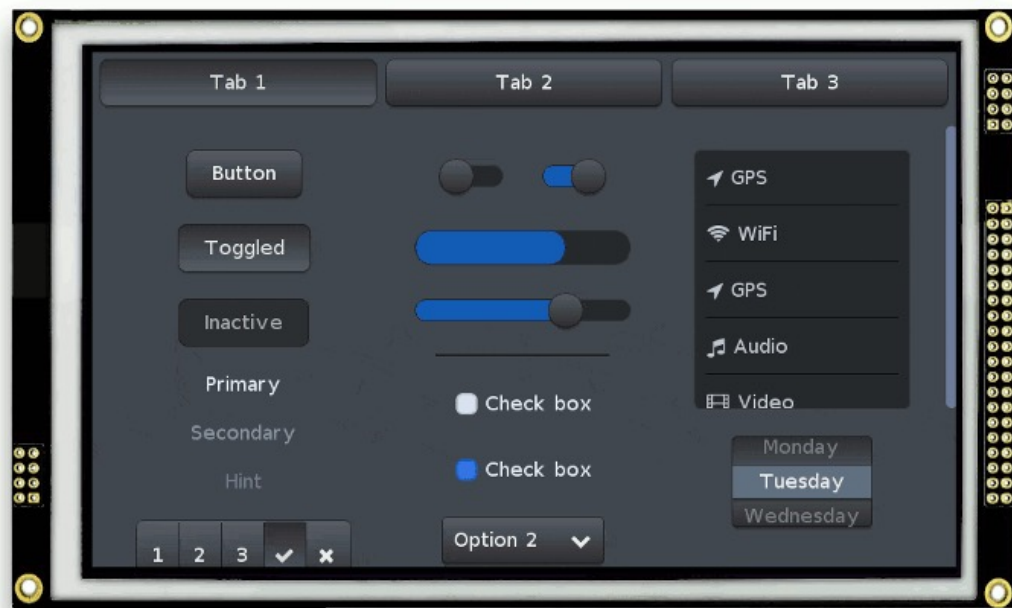
- 设计器
- 对象库、模板库、例程



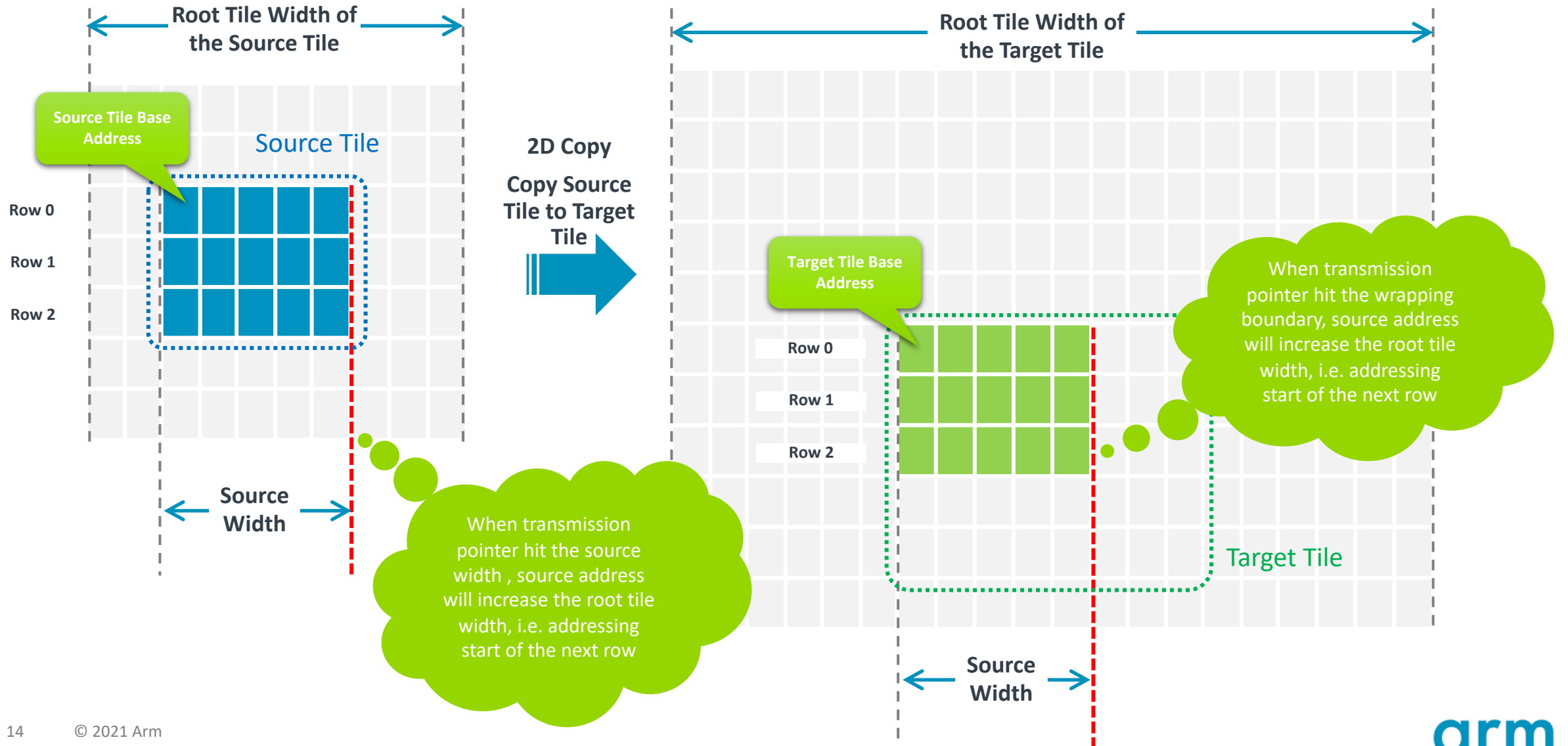
Arm-2D是“标准显卡驱动”
用于加速和支持GUI

MCU上实现2D图形界面的技术可行性

- 2D图形处理所需的常见技术非常成熟
 - 早在6502时代就已经登峰造极
 - 基于贴图技术的精美界面设计在HTML时代已经成熟
 - 核心算法对算力要求不高



贴图的基本原理



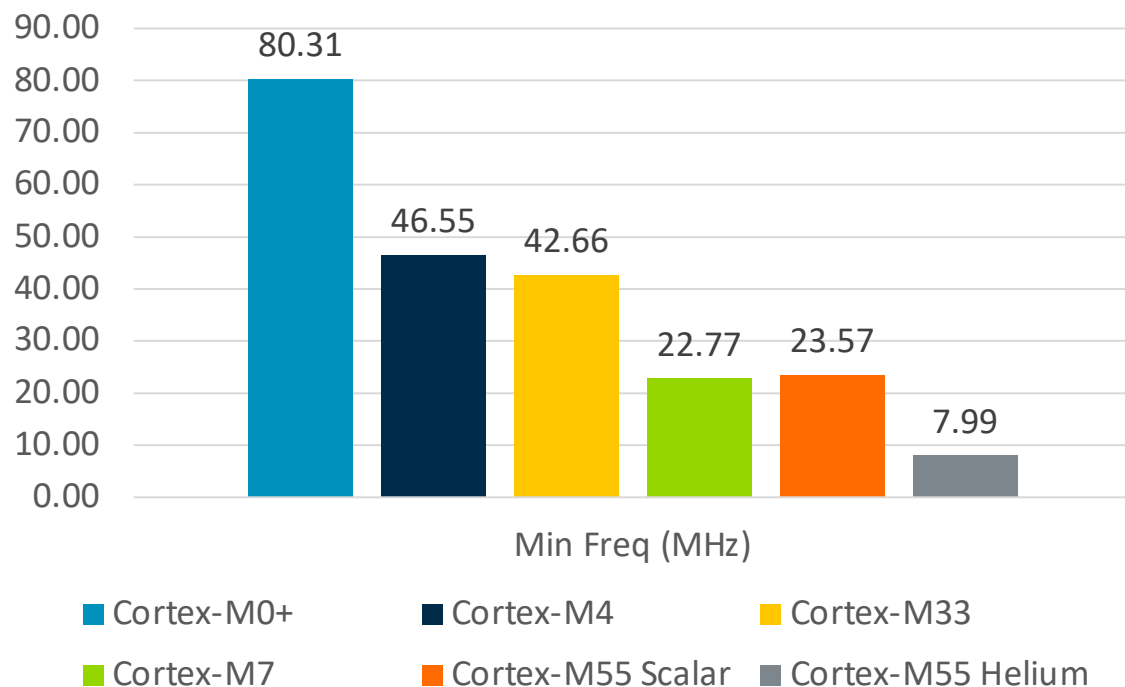
典型负载参考例程

- 背景图层:
 - 在黑色幕布上填充CMSIS图标.
 - 每隔4秒钟切换一下填充时所使用的镜像模式（无镜像、X镜像、Y镜像、XY镜像）
- 浮动的图片:
 - 通过一个会飘出显示范围的图片来展示Region切割功能以及对负数坐标的支持.
- 一个浮动的半透明红色色块
- 一个浮动、带有纹理填充的半透明绿色色块
 - 填充的太阳使用了抠图技术
- 一个浮动的太阳图标
 - 它用来展示如何使用抠图技术来实现常见的指针和图标功能
- 一个典型的“系统忙”指示
 - 这是使用“支持透明的抠图”技术实现的
 - 白色的小圆圈实际上是一个黑底白面的小图片 (◼)。

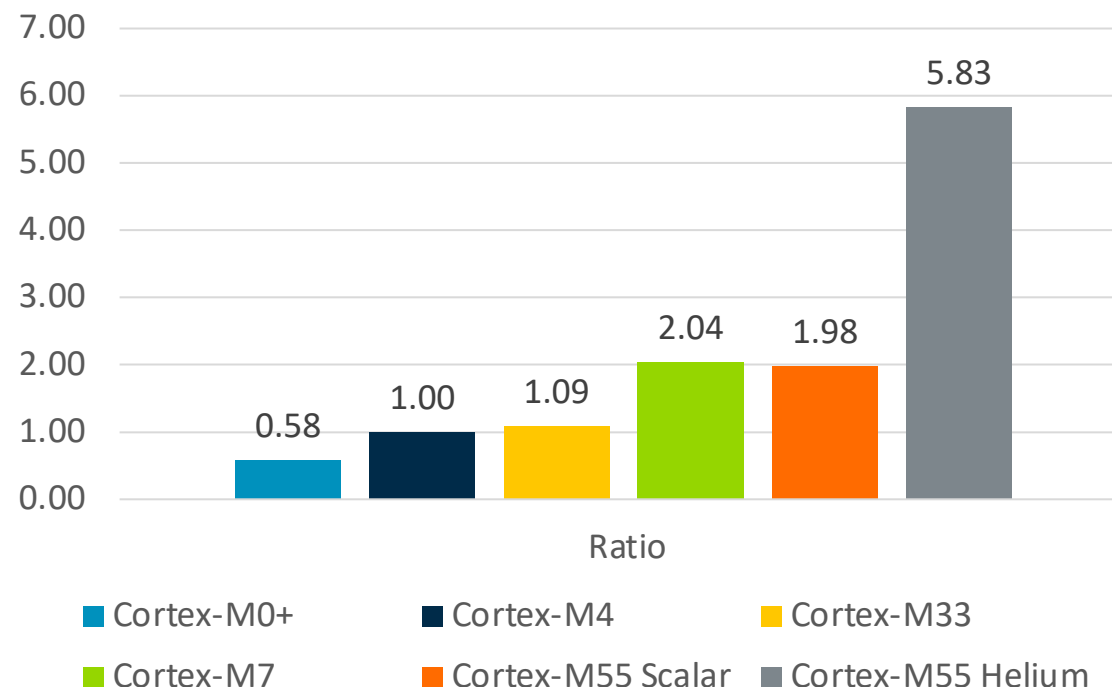


常见Cortex-M处理器的性能对比

达到 30 FPS 时所需的最小系统频率 (MHz)



以Cortex-M4为参考的性能比例



嵌入式GUI的灯下黑

- 常见的嵌入式GUI都注意到了MCU资源受限的特点
- 很多主打小资源MCU的GUI仍然无法覆盖很大一部分常用MCU的资源范围
 - Flash: 32K ~ 64K
 - SRAM: 4K ~ 32K
- 支持Partial Framebuffer成为必须
 - 用户自己实现PFB非常困难

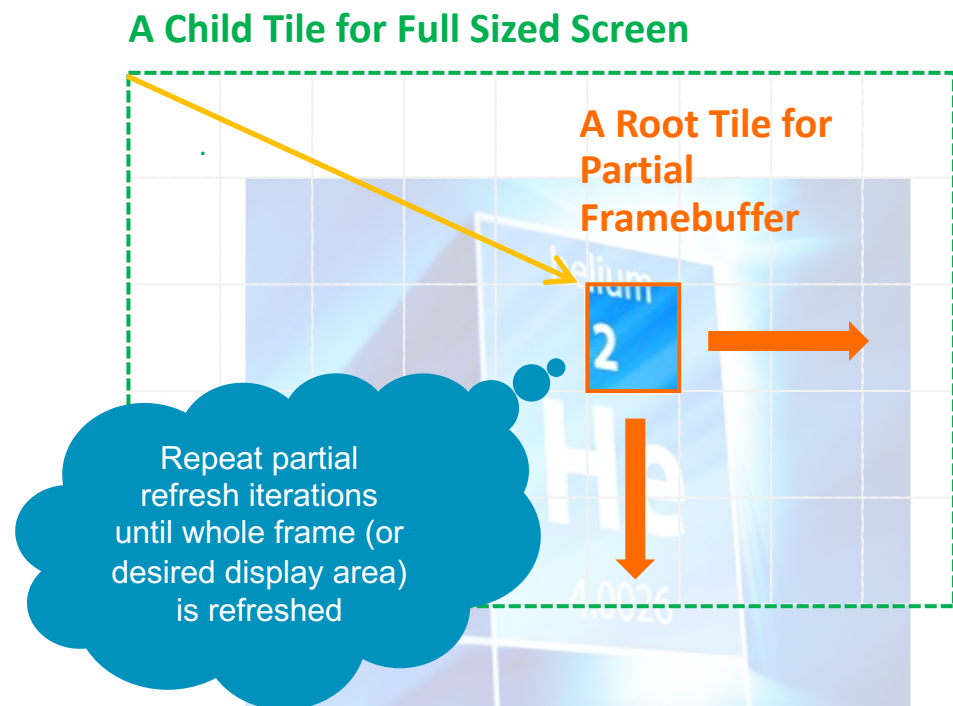
Requirements

- ✓ 16,32 or 64 bit microcontroller or processor
- ✓ > 16 MHz speed is recommended
- ✓ Flash/ROM: > 64 kB (180 kB is recommended)
- ✓ RAM: 8 kB (24 kB is recommended)
- ✓ 1 Frame buffer: in MCU, external RAM, or display controller
- ✓ Graphics buffer for LVGL: > "Horizontal resolution" pixels (1/10 "Screen size" is recommended)
- ✓ C99 or newer compiler
- ✓ Basic C (or C++) knowledge: pointers, structs, callbacks

Arm-2D也意味着“从无到有”
为小资源MCU 实现精美GUI界面提供了可能

简单通用的PFB支持

- 背景:
 - 很多单片机的SRAM通常都不是很大 (4K ~ 32K)
 - 一个常见的低成本320*240 的16位色LCD屏幕就要消耗150KB RAM作为显示缓冲.
 - 很多嵌入式应用对显示的帧率要求都不高.
- Arm-2D为所有Cortex-M处理器都提供了PFB
 - 高层API可以假装拥有完整的Framebuffer
 - 哪怕你只有8*8大小的豆腐块 (PFB) 你也可以支持任意大小的大屏幕.
 - 实际上PFB的大小和形状可以任意.
 - 用PFB来刷新的屏幕大小没有限制
 - 提供傻瓜式的PFB辅助模块
 - 支持局部刷新
 - 要刷新的一连串目标区域由用户提供.



常见的2D图形操作分类

操作类型	分类	性能要求	重要性	SW/HW	ACI	MVE	DMA	Comments
Tile Operations	基础	低	标配	X		X	X	图片的拷贝和纹理的填充
Colour Space Conversion	典型	看情况	一般都有	X	X	X	X	颜色格式的转换
Drawing	典型	还好	一般都有	X		X		画点、画线、颜色填充等等
Alpha Blending	典型	较高	一般都有	X	X	X	X	透明效果（抠图、透明蒙版等等）、图层合成等等
Mirroring	典型	低	一般都有	X		X		镜像
Rotation	高级功能	吃力	选配	X	X	X		图片的旋转
Zooming/ Stretching	高级功能	吃力	选配	X	X	X		缩放和拉伸
Filtering	高级功能	吃力	比较少见	X	X	X	X	各种滤镜效果（抗锯齿等等）

The ARM logo is displayed in a white, lowercase, sans-serif font. The background of the slide features a complex, glowing blue circuit board pattern with various traces and components, set against a dark blue gradient. Small white plus signs are scattered across the background.

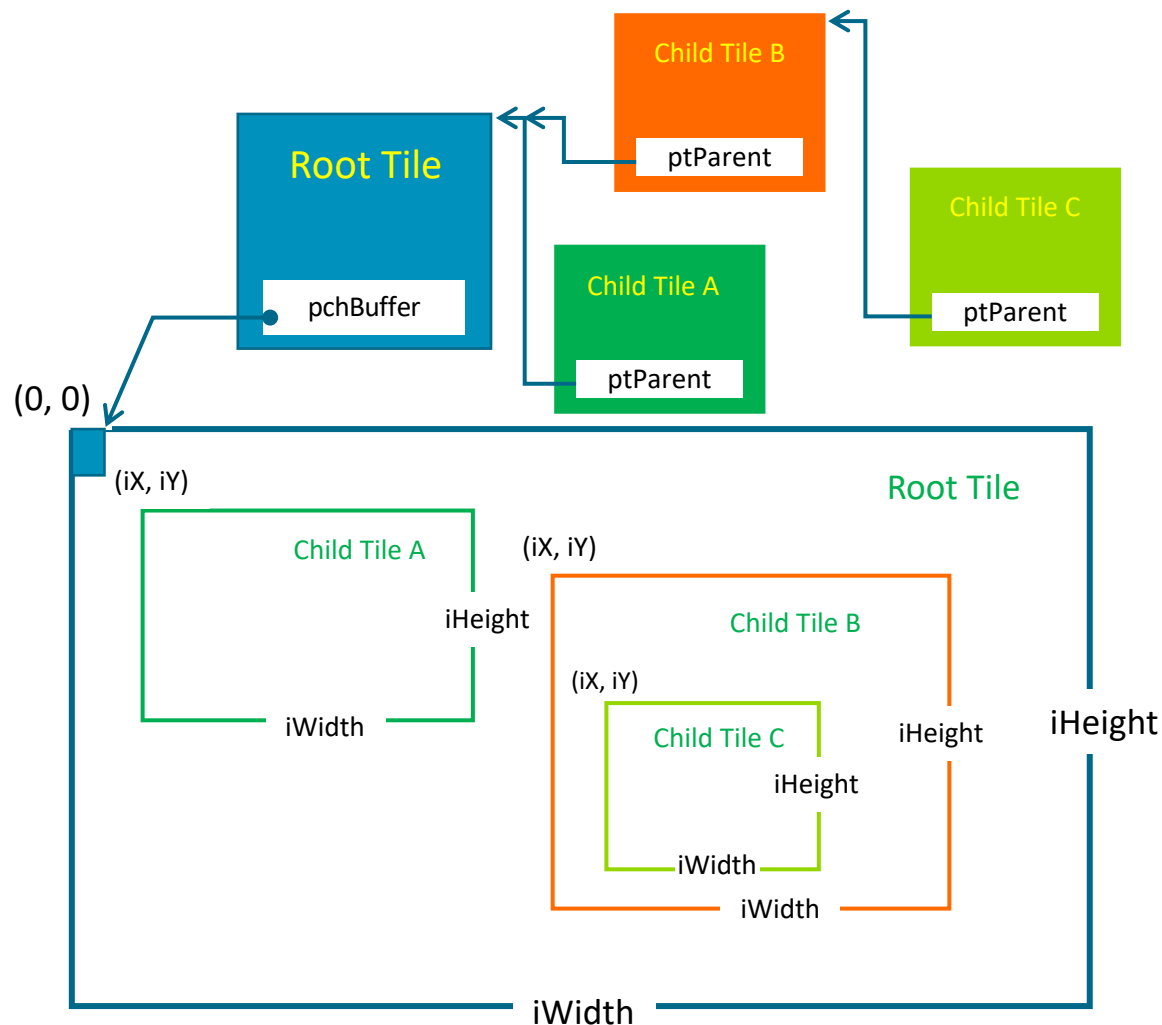
arm

Arm-2D的开发者模型

The Programmers' Model of Arm-2D

贴图 (Tile)

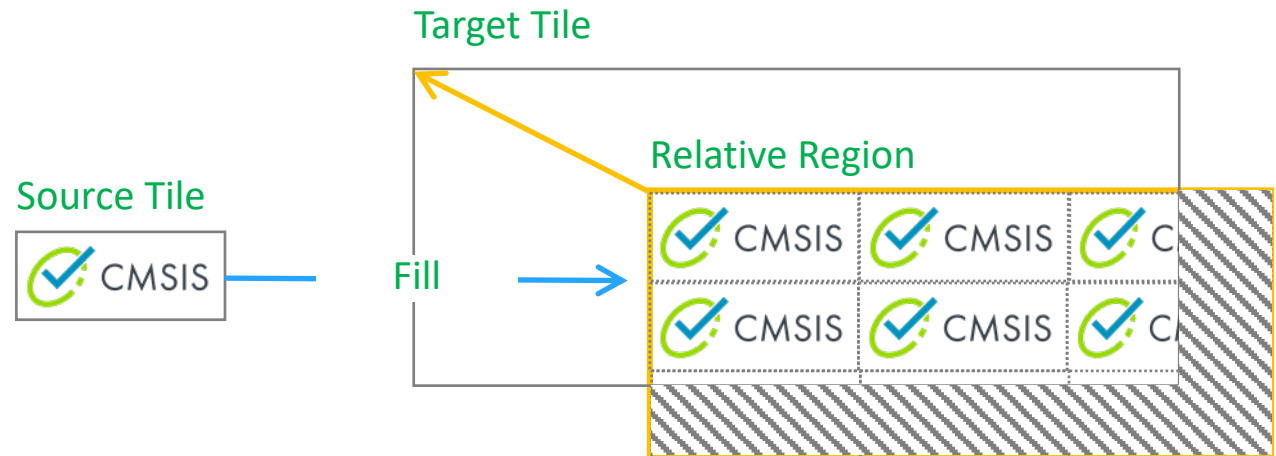
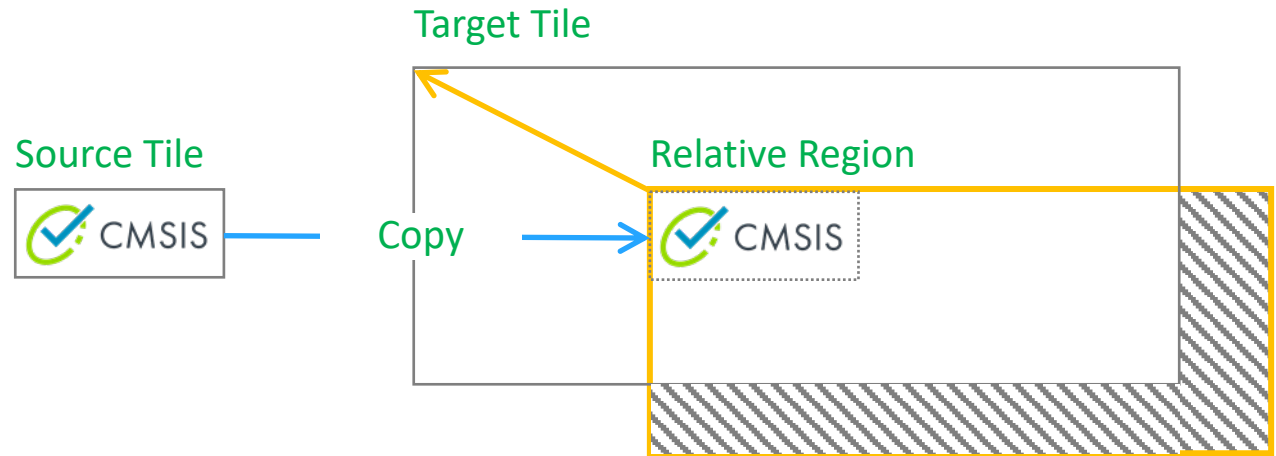
- 贴图是Arm-2D操作的基本单位。它可以用来描述所有的资源：
 - 图片
 - 显示缓冲区
 - 字体
 -
- 根贴图 (Root Tile) 意味着它真正的拥有资源
 - 资源的尺寸信息
 - 资源缓冲区的地址
 - 像素的色彩信息
- 在任何贴图的基础上都可以派生出子贴图。
 - 子贴图中的位置信息标 (Location) 注了自己在父贴图
中的相对坐标 (Relative Coordinates)
 - 包含了一个指向父贴图的指针



注意：上述所有的贴图（包括子贴图）实际上共享了同一片缓冲区。

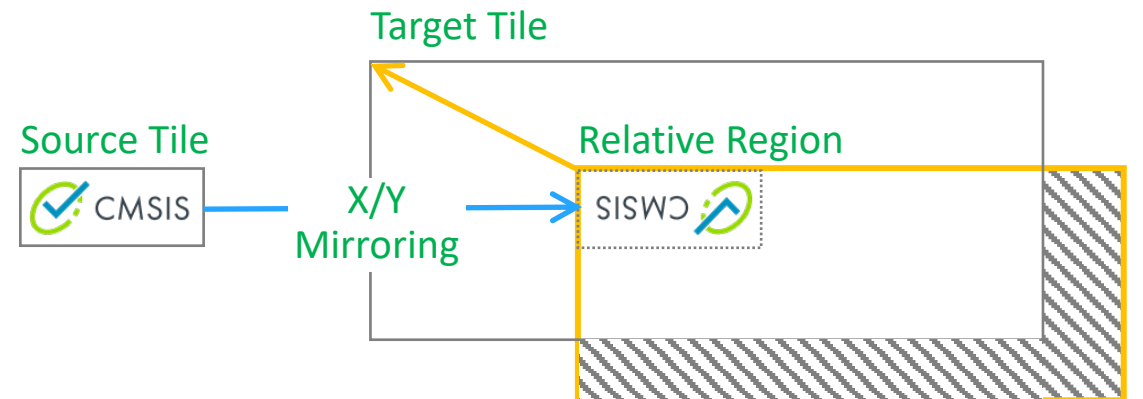
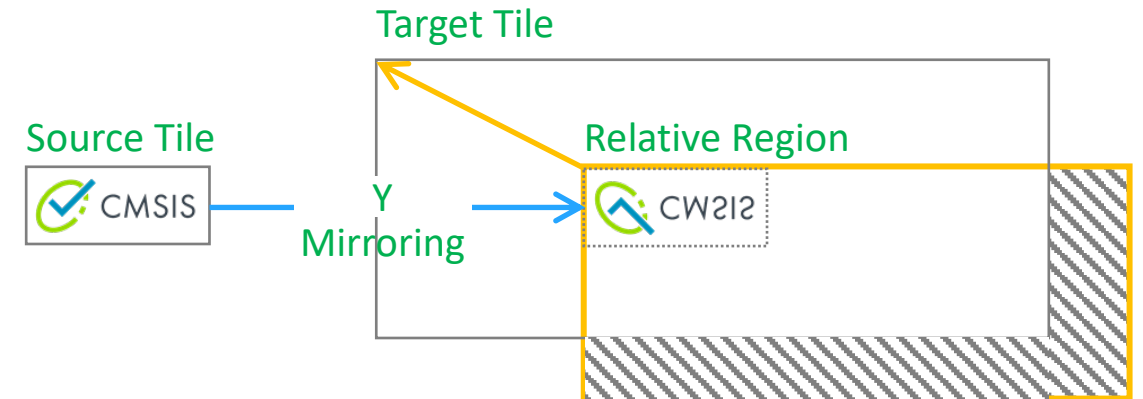
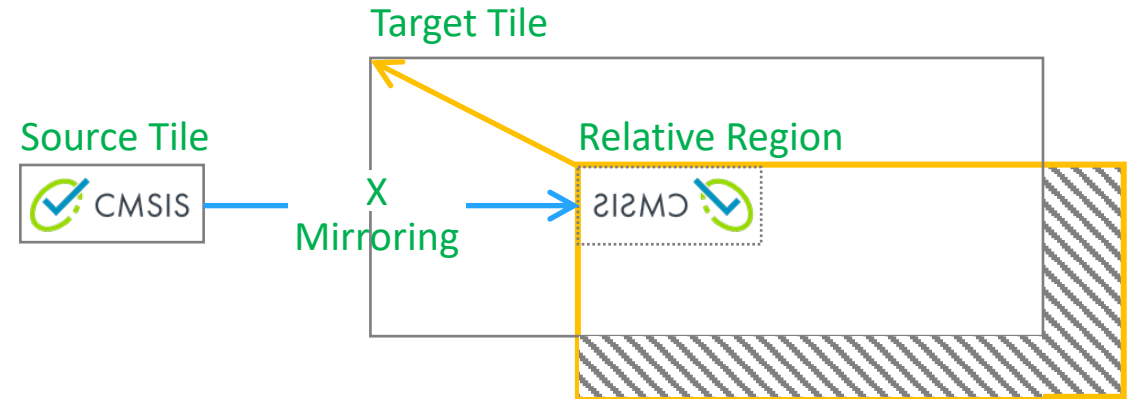
贴图的拷贝和填充

- arm_2d_rgb16_tile_copy / arm_2d_rgb32_tile_copy
- Input:
 - Source Tile
 - Target Tile
 - Relative Region in Target Tile
 - 拷贝还是填充
- Output:
 - An Updated Target Tile



带镜像模式的拷贝和填充

- arm_2d_rgb16_tile_copy / arm_2d_rgb32_tile_copy
- Input:
 - Source Tile
 - Target Tile
 - Relative Region in Target Tile
 - 镜像模式
- Output:
 - An Updated Target Tile



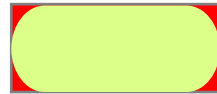


- arm_2d_rgb16_tile_copy_with_colour_masking/
arm_2d_rgb32_tile_copy_with_colour_masking
- Input:
 - Source Tile
 - Target Tile
 - Relative Region in Target Tile
 - Mask colour
- Output:
 - An Updated Target Tile

Mask Colour

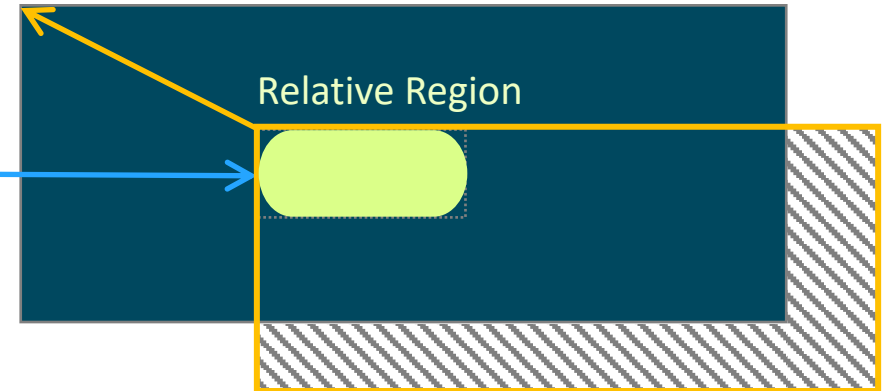


Source Tile



Copy

Target Tile



arm

实际操作演示

Tile 的数据结构

- 属性信息
- Region信息
- 指针区域

```
typedef struct arm_2d_tile_t arm_2d_tile_t;
struct arm_2d_tile_t {
    implement_ex(struct {
        uint8_t    bIsRoot           : 1;    //!< is this tile a root tile
        uint8_t    bHasEnforcedColour : 1;    //!< does this tile contains enforced colour info
        uint8_t    : 6;
        uint8_t    : 8;
        uint8_t    : 8;
        arm_2d_color_info_t    tColourInfo;    //!< enforced colour
    }, tInfo);

    implement_ex(arm_2d_region_t, tRegion);

    union {
        /*! when bIsRoot is true, phwBuffer is available,
        *! otherwise ptParent is available
        */
        arm_2d_tile_t    *ptParent;
        uint16_t         *phwBuffer;
        uint32_t         *pwBuffer;
        uint8_t          *pchBuffer;
    };
};
```

arm

提问环节

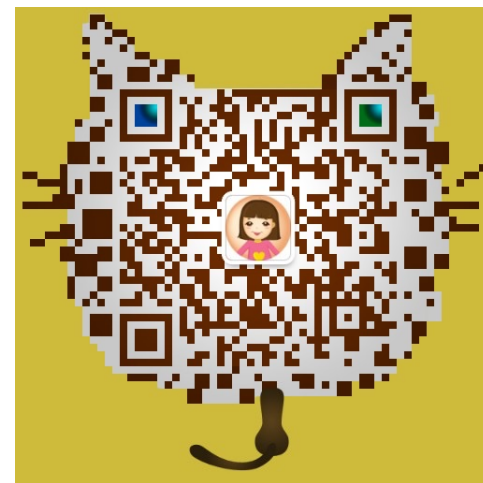


极术社区

更多极术公开课请关注极术社区



关注极术社区微信号



添加小助手加入技术群

arm

Thank You

Danke

Gracias

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה

arm

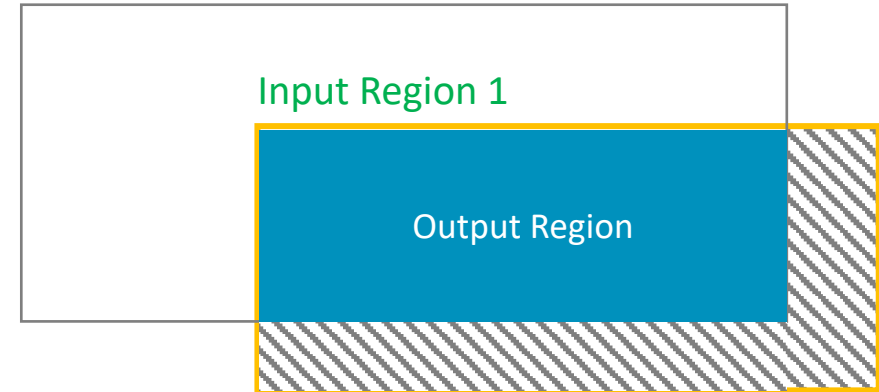
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

Region求交集

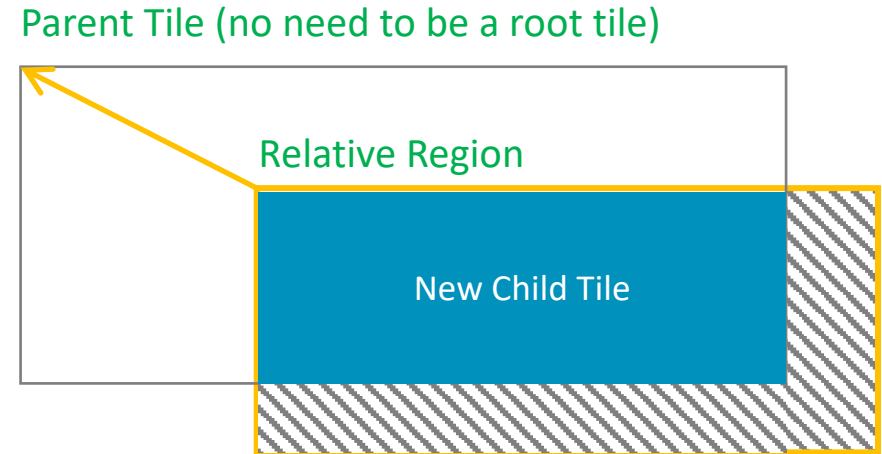
- `arm_2d_region_intersect`
 - Input:
 - Input Region 0
 - Input Region 1
 - Output:
 - A clipped region

Input Region 0



如何动态生成一个子贴图

- `arm_2d_tile_generate_child`
 - Input:
 - Parent Tile
 - A relative region of the given parent tile
 - Output:
 - A new child tile with clipped region



如何找到根贴图

- `arm_2d_tile_get_root`
 - Input:
 - A given tile
 - A buffer to store the valid region info
 - Output:
 - The root tile

