ARM Cortex magú mikrovezérlők

13. Garfikus könyvtár

Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

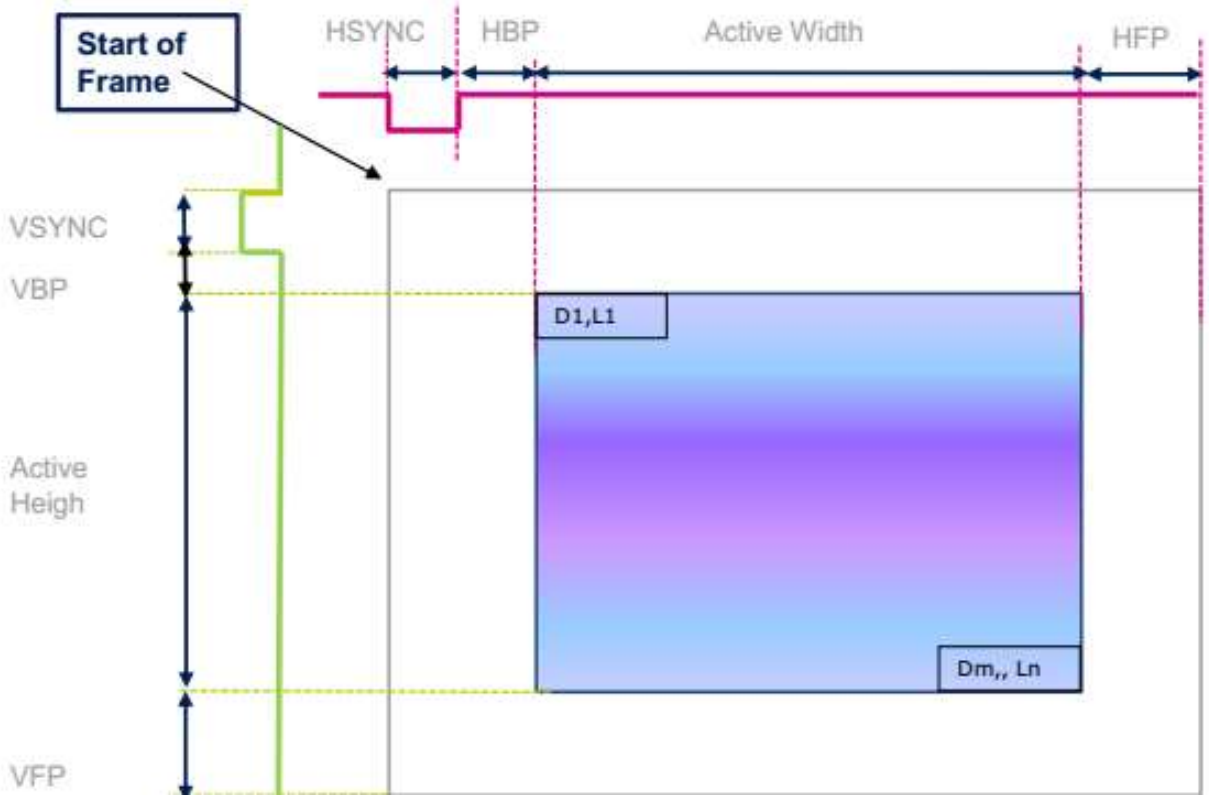
LCD kijelzők kezelése

LCD kijelzők kezelése

- Beépített kontrollerekkel rendelkező vezérlők kezelése egyszerű
  - Tipikusan SPI-on keresztül lehet programozni
  - Pixelek memória címekre vannak leképezve
- Beépített kontroller nélkül
  - Bonyolult, folyamatos nagysebességű frissítés
  - Nagy memória igény
  - Hardware támogatás nélkül elképzelhetetlen

LCD vezérlés

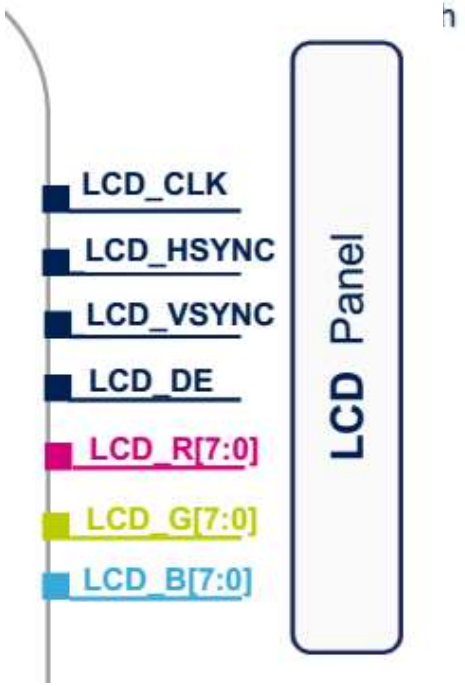
- TFT LCD controller



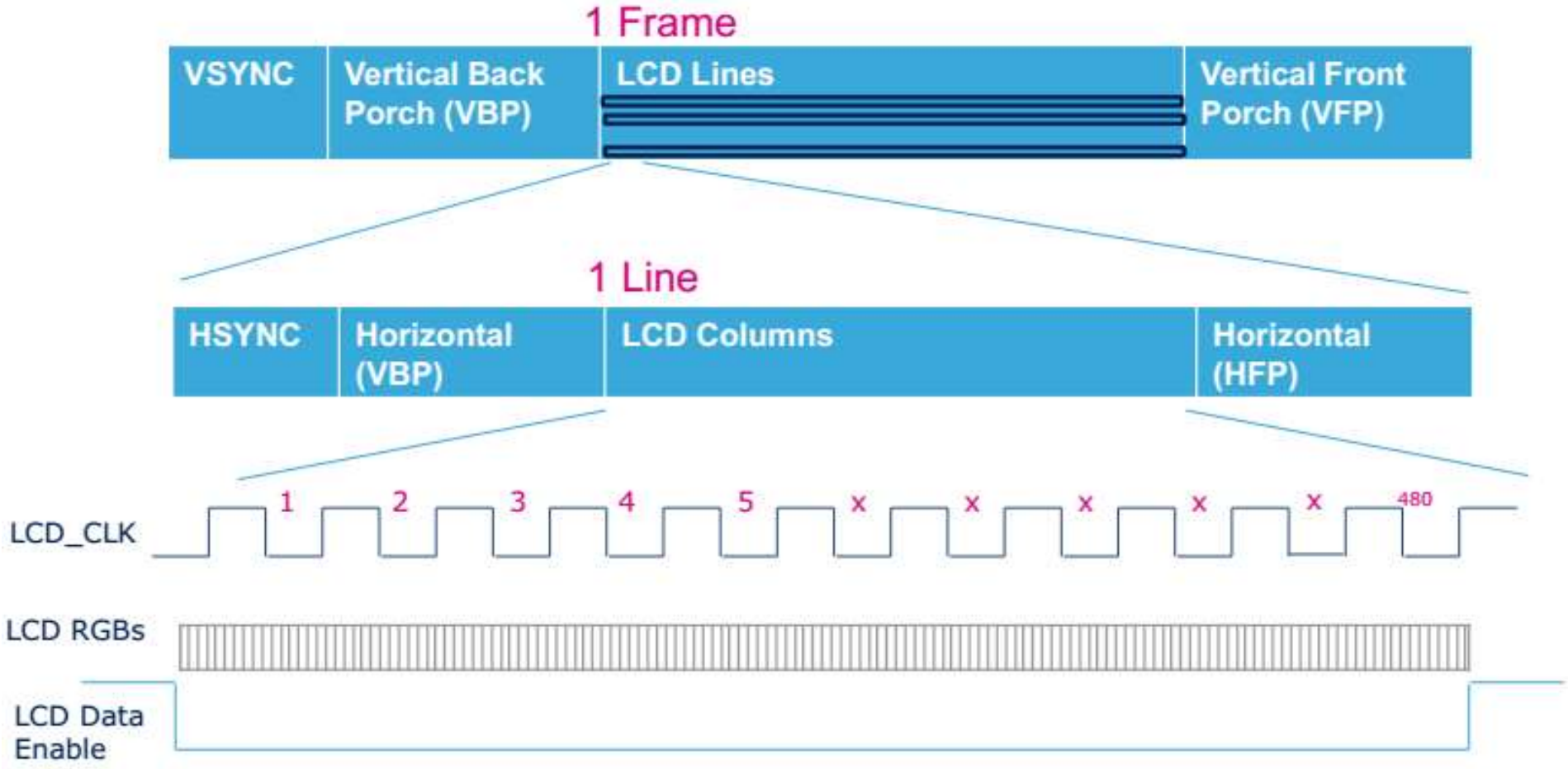
VBP: Vertical Back porch

VFP: Vertical Front porch

HBP: Horizontal Back porch



LCD vezérlés

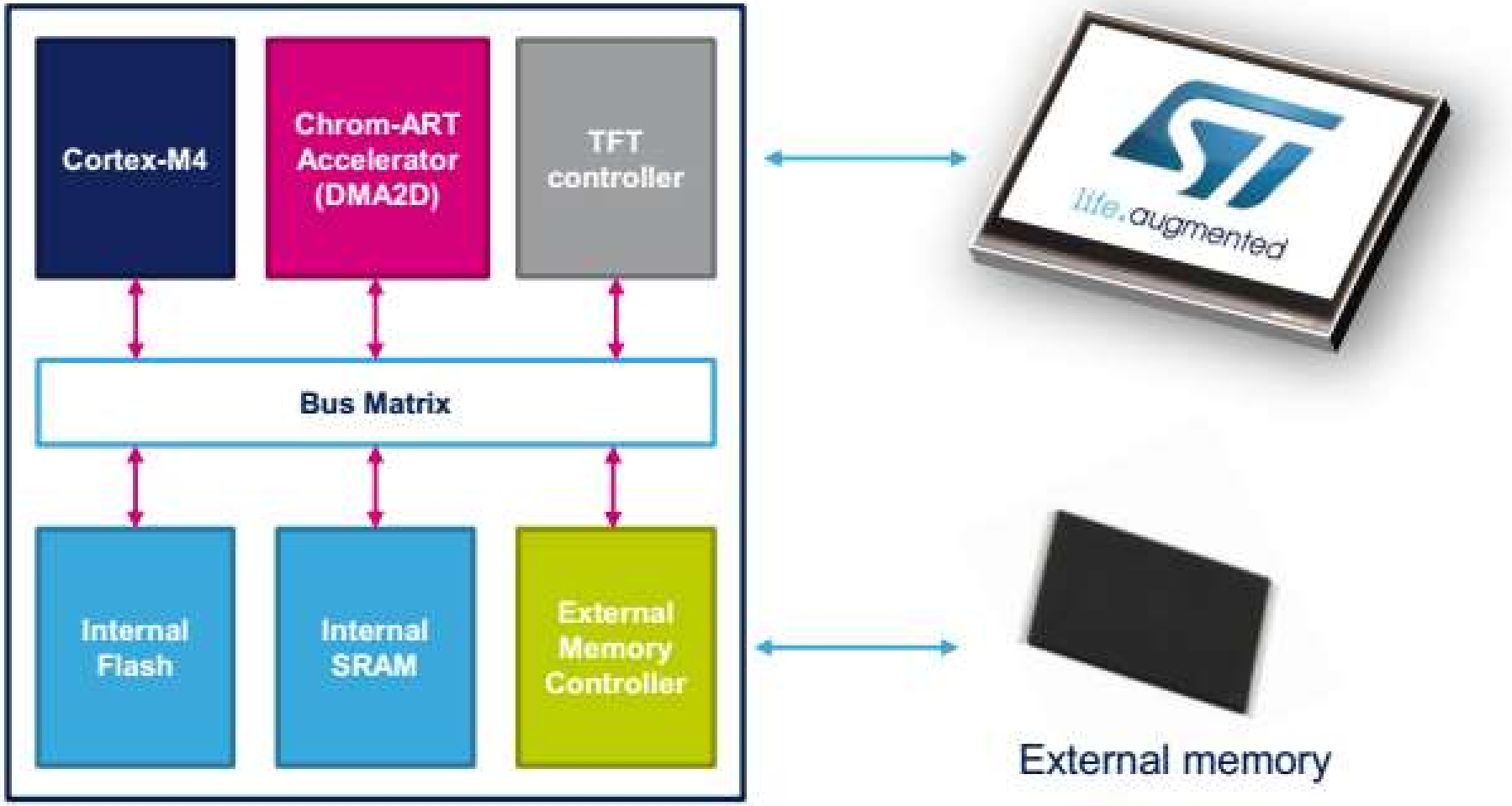


LCD vezérlés memória igénye

- Sokszor nem elég a táblázatban leírt memória, mert dupla bufferelés szükséges. Egy a jelenlegi, ami kijelzés alatt van egy pedig ami a következő képet tartalmazza.

Panel Resolution	Total Pixel	bpp (Bit per pixel)	Required memory (KB)
320x240 (QVGA)	76.8K	16bpp	153.6
		8bpp	76.8
480x272 (WQVGA)	130.5K	16bpp	216.1
640x480 (VGA)	307.2K	16bpp	614.4
800x600 (SVGA)	480K	16bpp	960

STM32f429 Grafikus támogatása

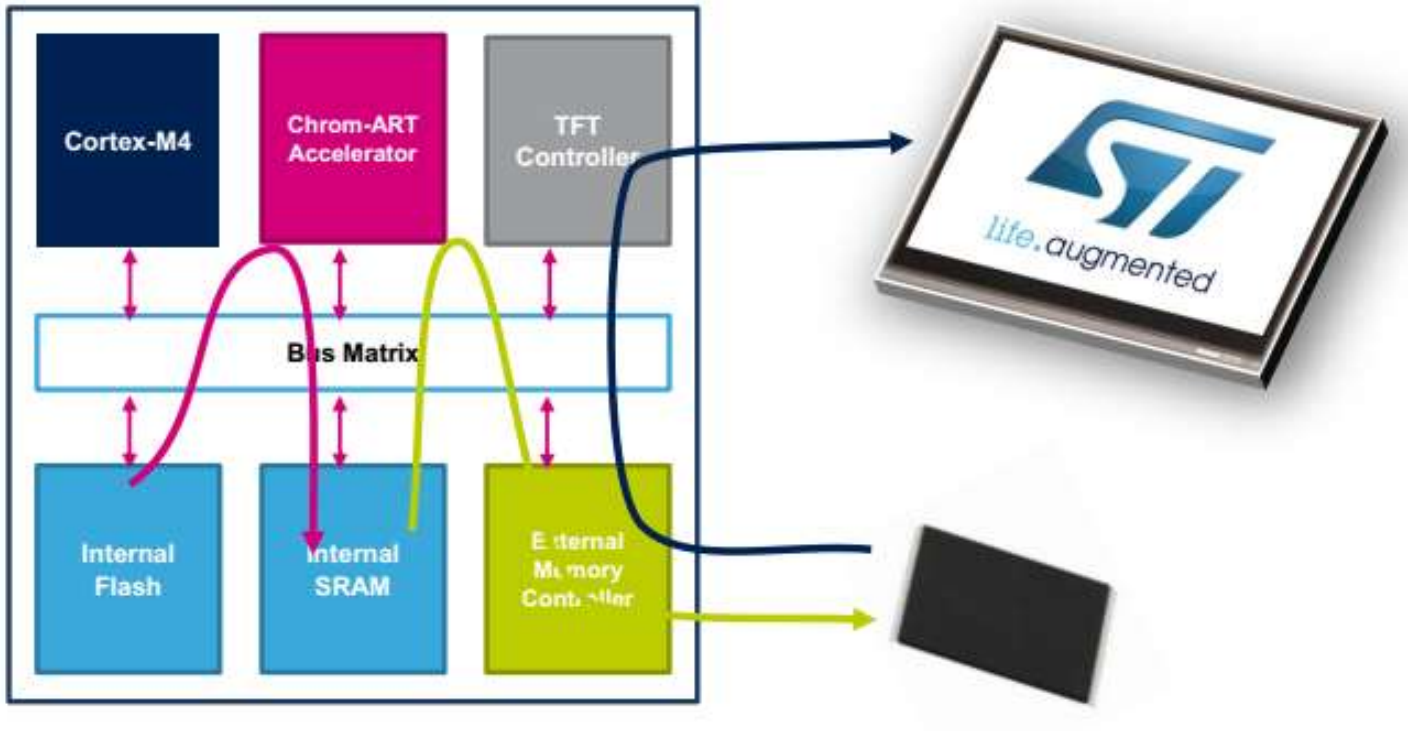


STM32f429 Grafikus támogatása

- 3 különböző órajel forrást használ:
  - memória interface, regiszter, LCD vezérlés
- Programozható LCD vezérlés időzítések V, H sync
- 24 bites RBG szín megjelenítésre is képes
- Max 800x600-as kijelző
- Több layer támogatása: 2 + background
- Ditherelés 2 bit / szin
- Áttetszőség beállítás

Chrom-ART Accelerator (DMA2D)

- CPU tehermentesítés:
  - Képmásolás, több ikon egymásra rakása különböző átlátszósággal



STM32 FMC controller: külső memória vezélés

- Külső memória támogatás:
  - SDRAM, SRAM, Flash
  - Max. 90 MHz, max. 32bites adatvonal

- SD RAM kezelés:
  - Állandóan frissíteni kell, mert elfelejti a tartalmát: automatikus frissítés programozható idővel
  - Energiamenedzsment támogatás csökkentett energiájú módokkal

Grafikus szoftver könyvtárak:

emWIN

Segger emWin Graphical Library

- Portolható grafikus library
- 8,16, 32 bites mikorvezérlőkre
- Oprendszer támogatás opcionális

- Minimum konfiguráció ~20kbyte Flash ~1k RAM

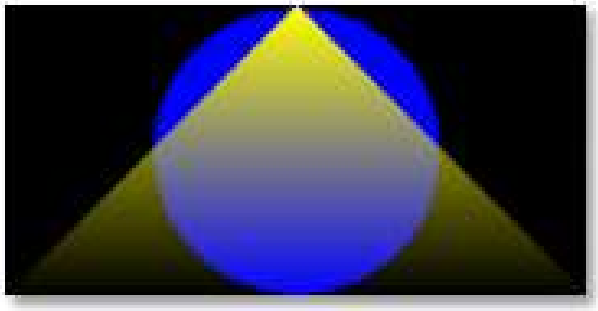
- Fizetős rendszer
  - Jó néhány gyártó ingyenesen elérhetővé teszi a saját környezetéhez

Alap 2D library

- Alap formák rajzolása, szöveg kiírása
  - Vonalak, négyszögek, háromszögek, sokszögek, körök
  - JPEG, PNG és bitmap
  - Az alapformák az előlapi színnel rajzolódnak, ami átállítható

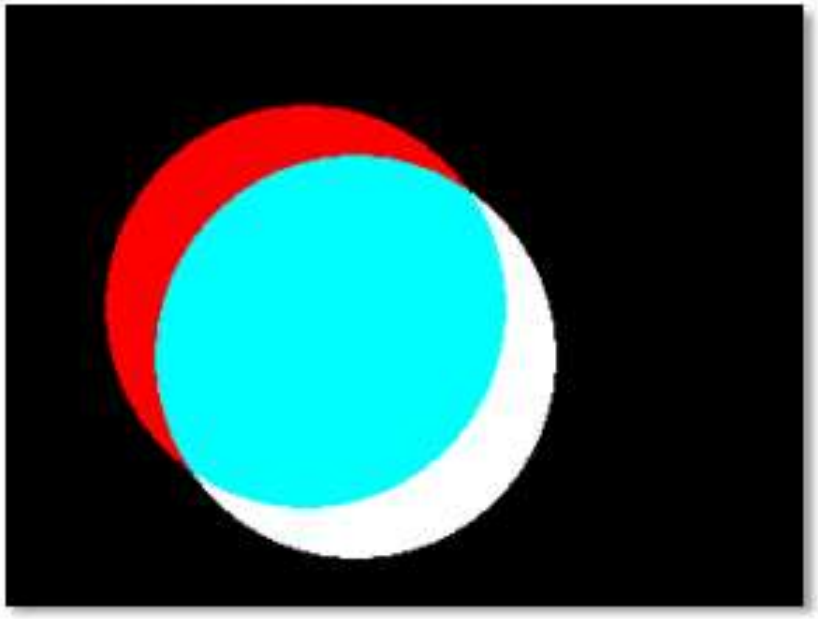
```
GUI_SetColor(GUI_RED);
```

  - Az objektumok átlapolódhatnak, és az alpha blending támogatott



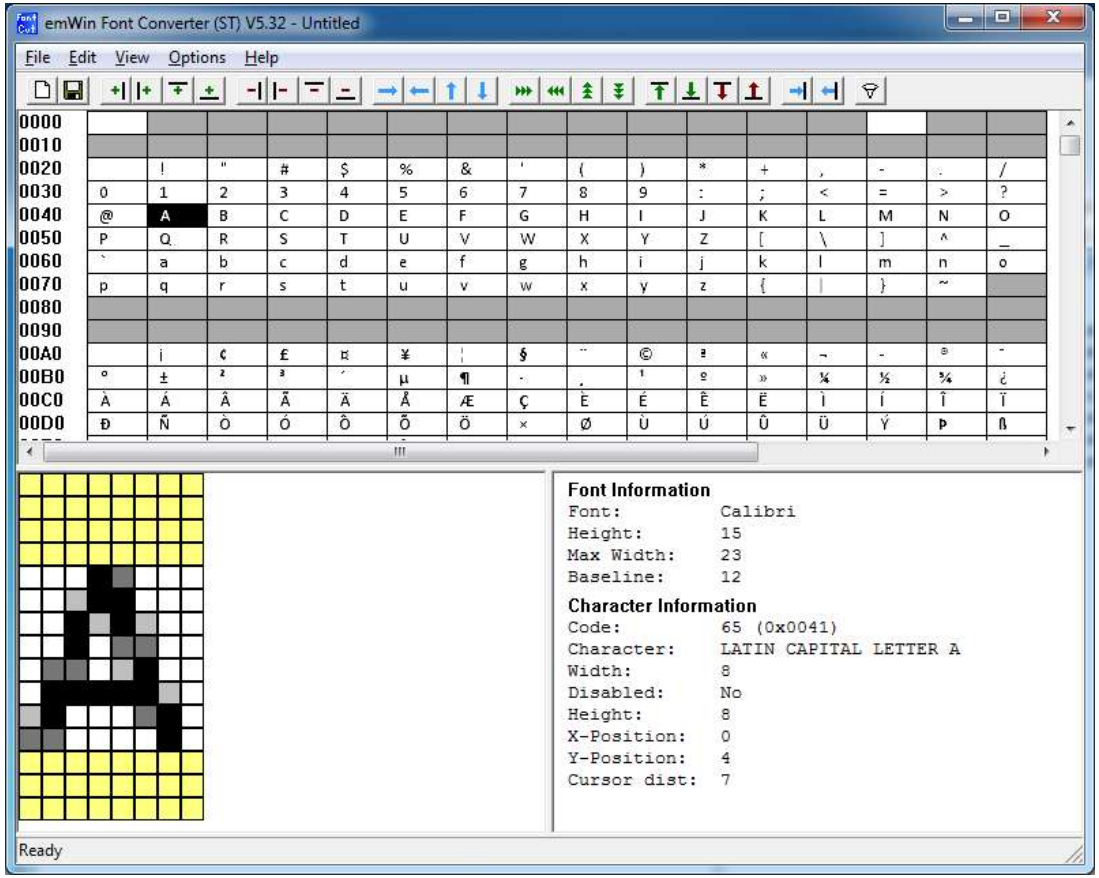
Alap 2D library példa

```
GUI_SetColor(GUI_RED);
GUI_SetDrawMode(GUI_DRAWMODE_NORMAL);
GUI_FillCircle(120, 120, 80);
GUI_SetDrawMode(GUI_DRAWMODE_XOR);
GUI_FillCircle(140, 140, 80);
```



Alap 2D library szöveg kiírása

- Külső programmal készíthető saját betű készlet
  - C file-okat exportálunk



Human interface

- Tipikusan valamilyen billentyűzetet, vagy pointer input eszközre vagy touch screen számítunk
- Ezek dedikált API függvényekkel vannak kezelve
  - **GUI_StoreKeyMsg()**: billentyűzet események kezelésére
  - **GUI_PID_StoreState():** Pointer Input Device esemény átadása (pl. egér esemény)
  - **GUI_TOUCH_StoreState():** Touch screen esemény átadása

Az emWin ablakozó rendszere

- Hierarchikus ablakkezelő rendszer
- Saját callback függvény rendszer támogatással
- Ez az alapja az összes Widget kezelésének

- Működhet oprendszer támogatással és nélküle is

Operációs rendszer nélküli működés

- Gyakorlatilag egy fő ciklusból folyamatosan hívni a ***GUI_Exec()*** függvényt.
- Nem szabad megfeledkezni a beviteli eszköz kezeléséről sem

```
void main (void) {
  HARDWARE_Init();

  /* Init software components */
  XXX_Init();
  YYY_Init();
  GUI_Init();          /* Init emWin */

  /* Superloop: call all software components regularily */
  while (1) {
    /* Exec all compontents of the software */
    XXX_Exec();
    YYY_Exec();
    GUI_Exec();        /* Exec emWin for functionality like updating windows */
  }
}
```

Operációs rendszerrel való működés

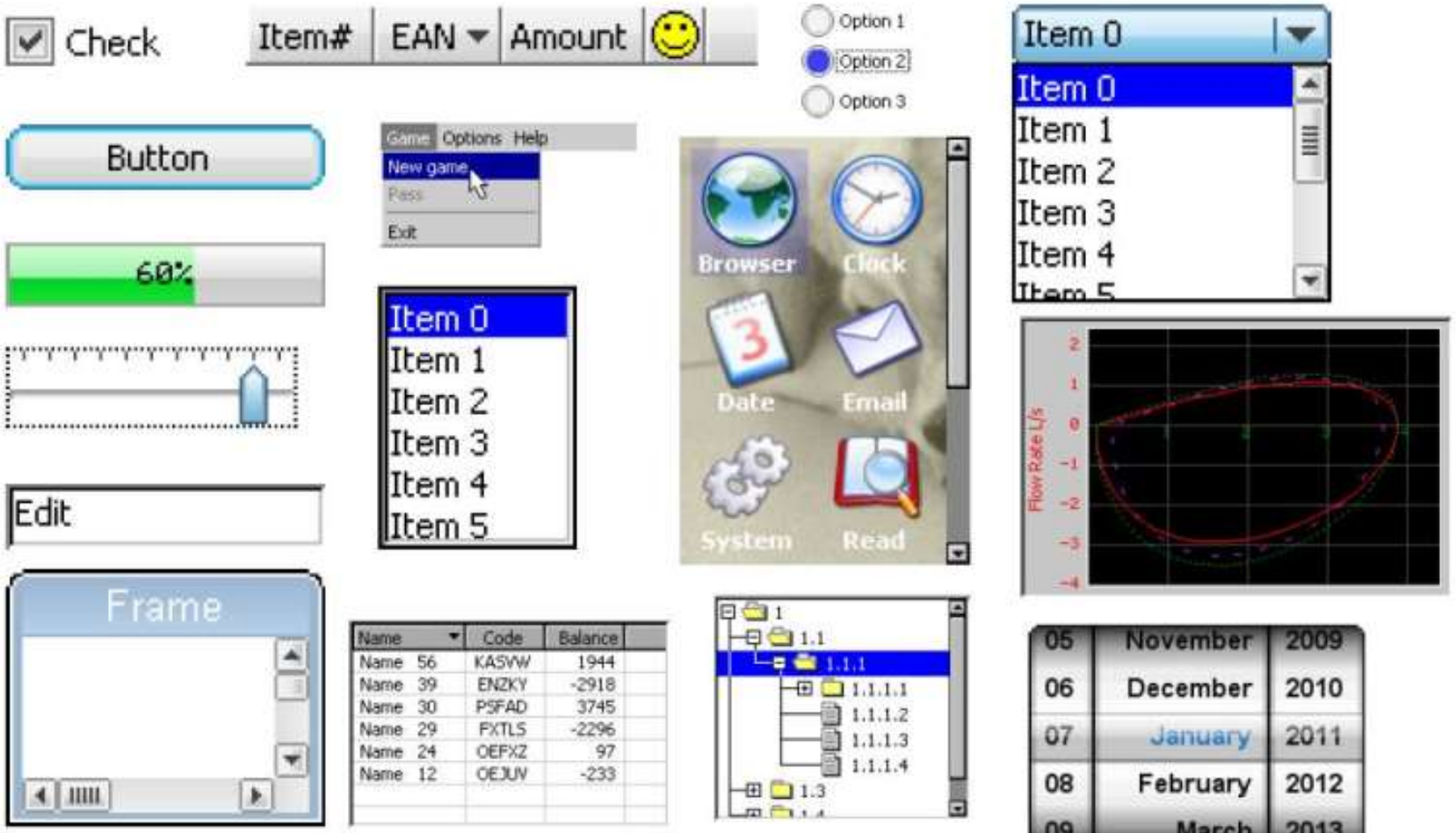
- Külön szálban a ***GUI_Exec(), GUI_Delay()*** függvények hívásával.
- A megfelelő define-al engedélyezni kell a konfigurációban (*GUIConf.h*)

```
#define GUI_OS        1     // Enable multitasking support
#define GUI_MAXTASK   5     // Max. number of tasks that may call emWin
```

```
/*********************************************************************
*
*          GUI background processing
*
* This task does the background processing.
* The main job is to update invalid windows, but other things such as
* evaluating mouse or touch input may also be done.
*/
void GUI_Task(void) {
  while(1) {
    GUI_Exec();            /* Do the background work ... Update windows etc.) */
    GUI_X_ExecIdle();      /* Nothing left to do for the moment ... Idle processing */
  }
}
```

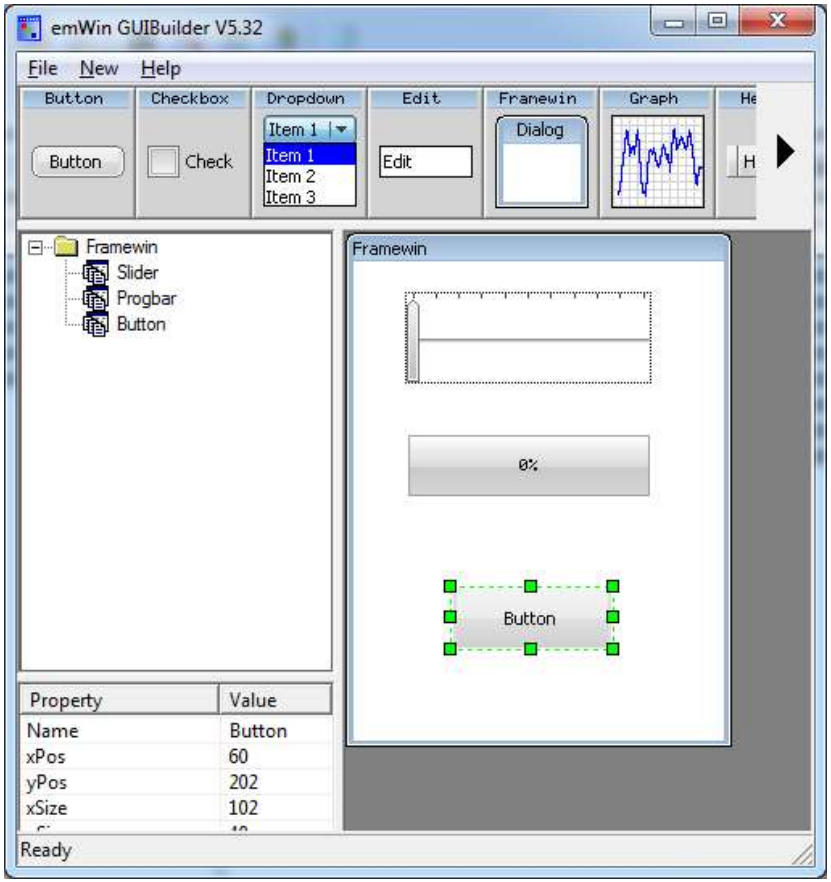
Widget: **Wi***ndow* + *Ga***dget**

- A legtöbbet használt kis grafikus kijelzők beavatkozók
- Rengeteg API függvénnyel



GUI builder

- Windows-os program amiben létrehozhatjuk könnyen a Widgeteket



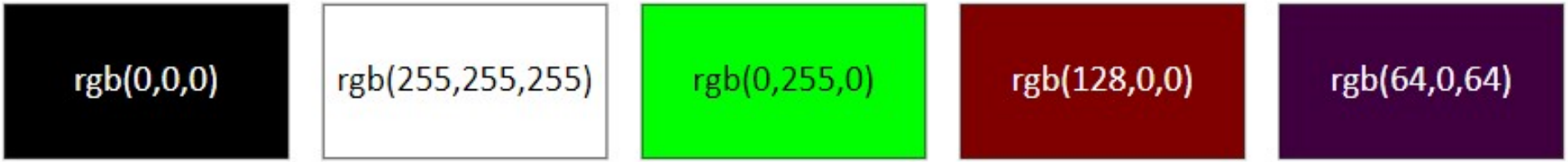
TouchGFX

TouchGFX összefoglalás

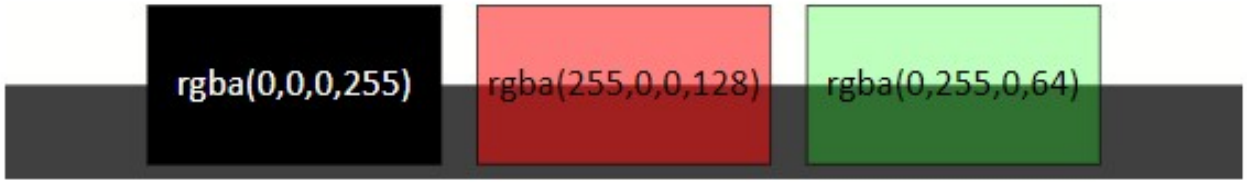
- Az STM32 fejlesztői kártyák közvetlen támogatása
- **TouchGFX Designer:** A TouchGFX-ben található, könnyen használható GUI-építő, vizuális megjelenés létrehozásához.
- **TouchGFX Generator:** Egy STM32CubeMX bővítmény, amelyben egy STM32 Cube projecthez generálható kód.
- **TouchGFX Engine:** A TouchGFX C++ keretrendszer, amely a felhasználói felületet vezérli. Kezeli a képernyőfrissítéseket, a felhasználói eseményeket és az időzítést.

Color formats

- Sokfajta színformátum támogatása a ARGB8888, től az 1-bit grey scale-ig. Tipikus formátumok: RGB565
  16 bit, 5 bit red, 6 bit green, 5 bit blue, ARGB8888 képeknek



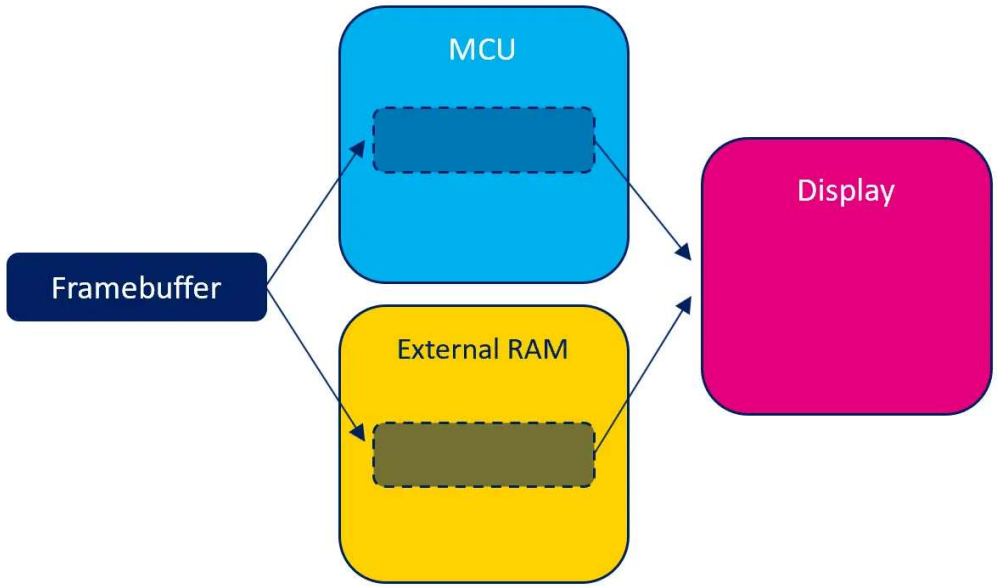
Some RGB colors



Some RGBA colors atop white and grey

Frame buffer handling

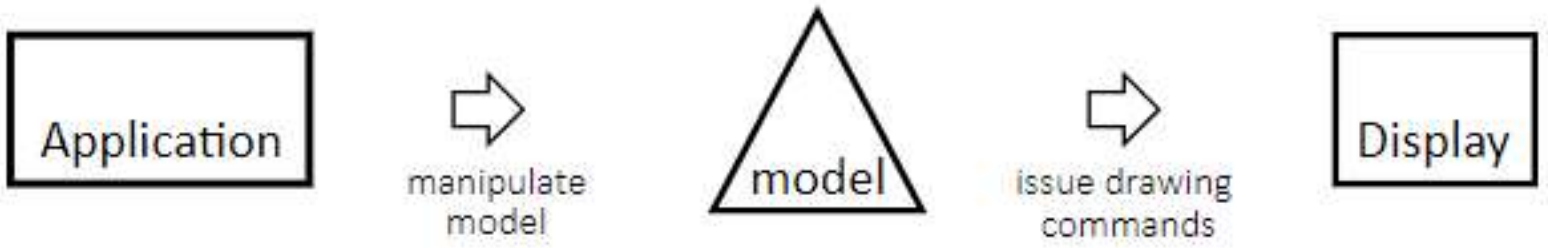
- Általában külső memóriából



Resolution (pixels)	Colors (bpp)	Calculation	Memory consumed (byte)
800x480	16 bpp	800 * 480 * 16 / 8	768.000 B
480x272	24 bpp	480 * 272 * 24 / 8	391.680 B
100x100	8 bpp	100 * 100 * 8 / 8	10.000 B

Retained mode graphics engine

- Retained mode graphics engine esetében a felhasználó az absztrakt grafikai modell-t manipulálja, és az engine gondoskodik a megjelenítésről a megfelelő időben.



Retained mode graphics

Retained mode graphics engine előnyei

Könnyű használhatóság: A retained graphics engine a legkönnyebben használható módszer. A felhasználónak csak az abszrakt grafikai modellekkel kell törődnie, azzal nem, hogy ezek hogyan lesznek megvalósítva.

Jó végrehajtási sebesség: TouchGFX analizálja a megjelenítési modellt, és csak azokat az objektumokat rajzolja ki, amelyek változnak, és ténylegesen szükséges a megjelenítésük. Rejtett objektumokkal nem foglalkozunk.

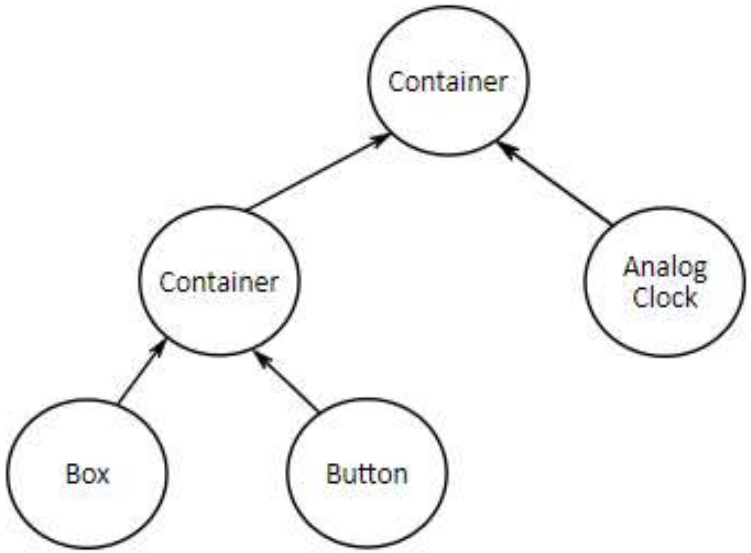
Állapot menedzsment: TouchGFX folyamatosan követi, hogy a kijelzés melyik része aktív.

Retained mode graphics engine hátányai

- **Memoria használat:** A megjelenítési modellek nyilvántartása elég sok memóriát igényel. A TouchGFX optimalizált memória használatában ez a kbyte tartományban marad.

A megjelenítési modell manipulása

- A „scene model” komponensekből áll.
- A komponensek leírása objektum orientált módon történik
- Az összes komponensnek egyetlen szülő komponense van. A megjelenítés modell általában fa formátumként írható le.



A tree of widgets

A megjelenítési modell manipulása

- Az alkalmazó szempontjából a megjelenés a megjelenítési modellben szereplő widgetek manipulásával változtatható meg..
- Például egy nyomógom, a pozitciójával, és a kijelzendő képével együtt a következő módon adható meg.

```
myButton.setXY(100,50);
myButton.setBitmaps(Bitmap(BITMAP_BUTTON_RELEASED_ID), Bitmap(BITMAP_BUTTON_PRESSED_ID));
add(myButton);
```

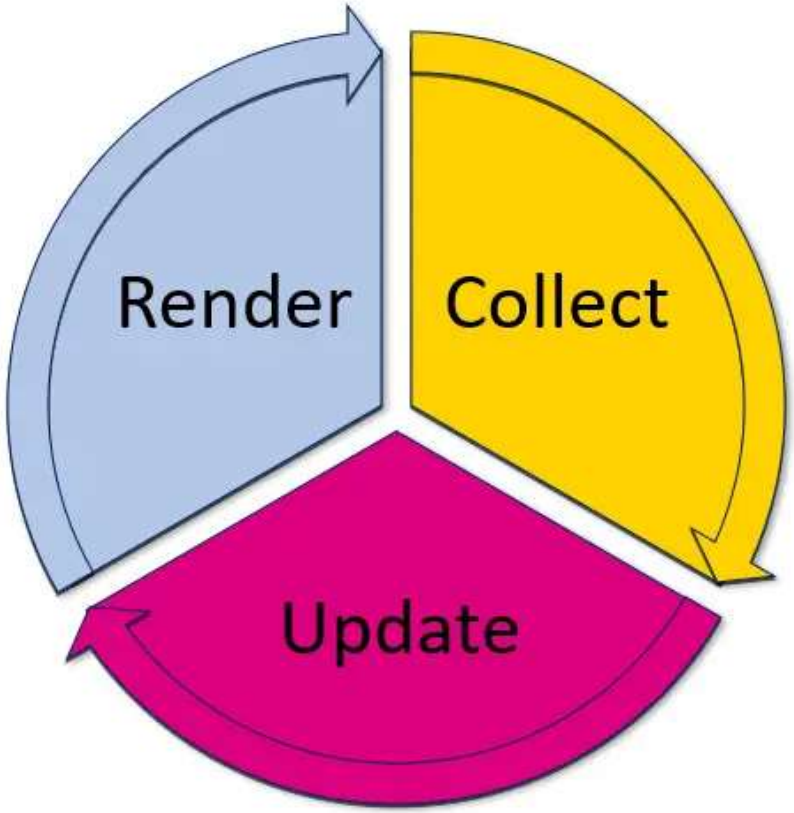
Renderelés

- A megjelenési modell renderelésekor a TouchGFX a rajzoló API-t használja (draw metódus). Ez a rajzoló API rendelkezik olyan grafikai primitívek rajzolásához szükséges módszerekkel, mint például dobozok, képek, szövegek, vonalak, poligonok, textúrázott háromszögek stb megjelenítése.
- A TouchGFX gomb widgetje például rendereléskor a drawPartialBitmap metódust használja a képek rajzolásához:

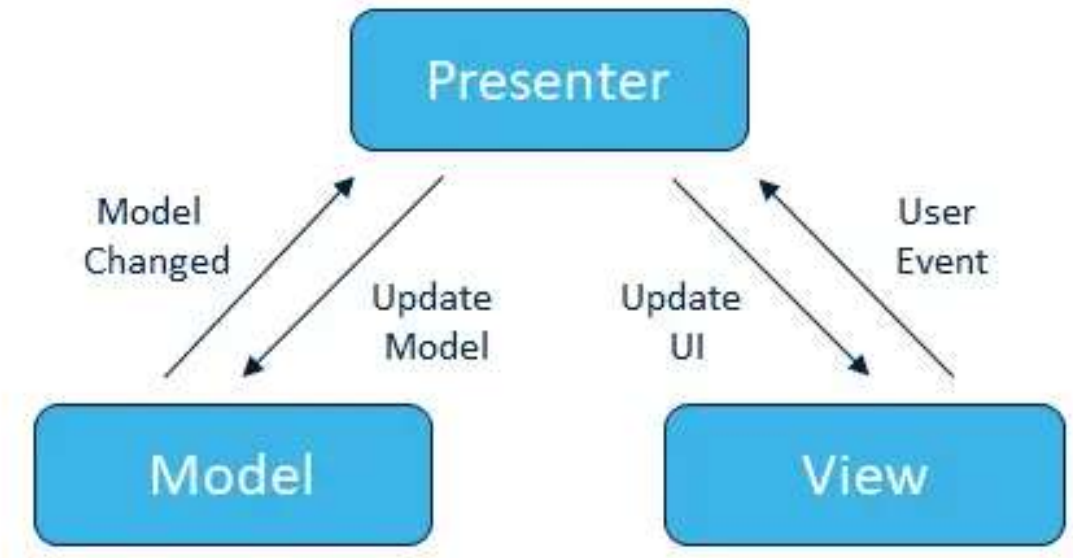
```
void Button::draw(const Rect& invalidatedArea) const
{
    // calculate the part of the bitmap to draw
    api.drawPartialBitmap(bitmap, x, y, part, alpha);
}
```

A touch GFX működése

- TouchGFX egy három fázisból álló végtelen ciklusban működik
  - **Események gyűjtése:** Események összegyüjtése a touch screen-ről, fizikai gombokról, backendtől
  - **A megjelenítési modell frissítése:** A begyűjtött eseményekre való reakció. A widgetek pozíciójának, tartalmának, kinézetének módosítása.
  - **Renderelés:** A grafikus display szükséges részeinek újrarajzolása

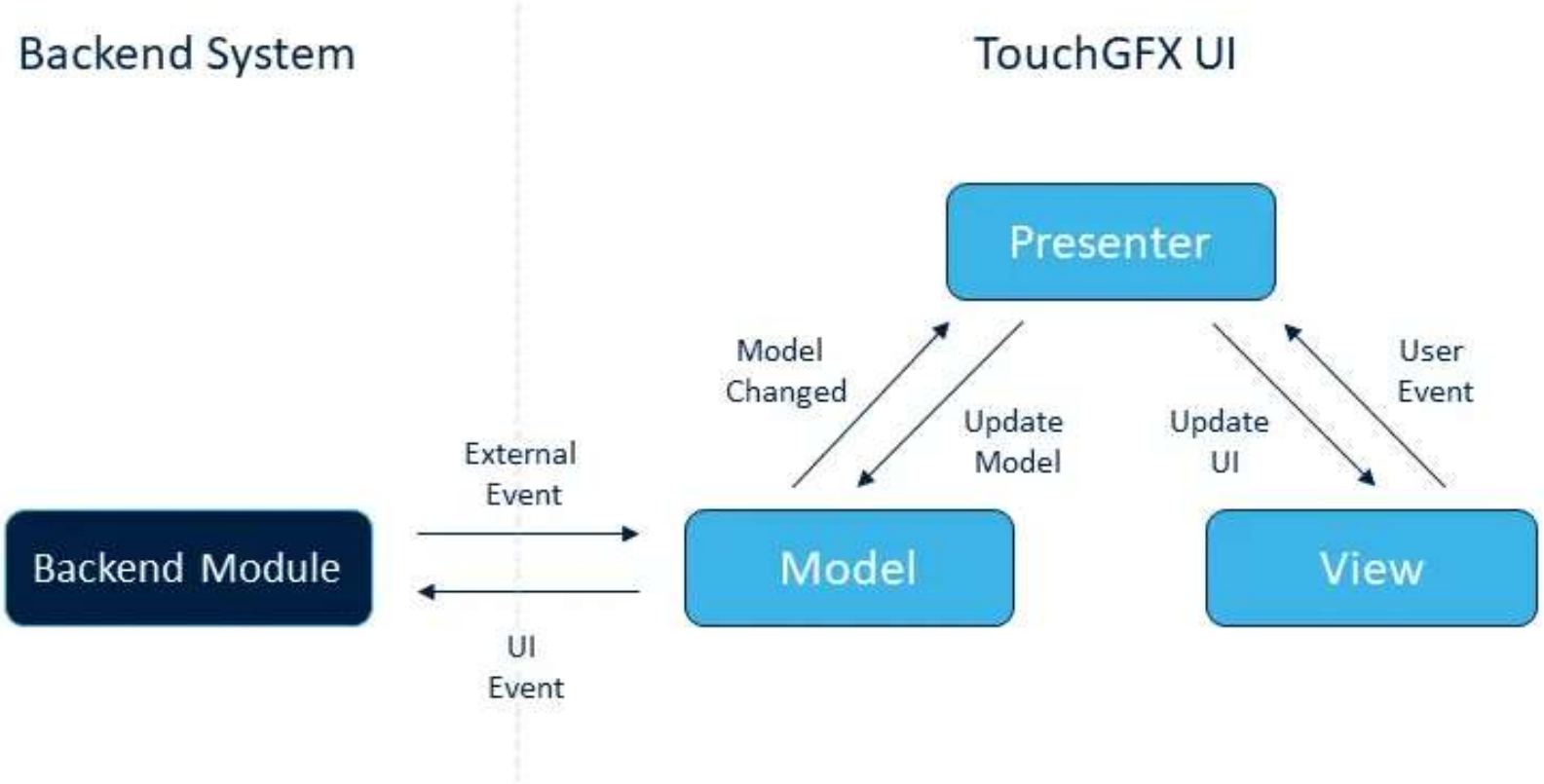


Kódfejlesztés a TouchGFX-re: Model-View-Presenter Design Pattern



- **A modell:** egy interfész, ami difiniálja a megjelenítendő adatot
- **A view:** egy passzíve interfész ami megjeleníti az adatot (a modellből) és a felhasználói eseményeket eljuttatja a presenterhez.
- **A presenter** A presenter frissíti a modell-t és a View-t. Fogadja az adatokat a modell-től, és továbbadja megjelenítésre a View-nak.

Kódfejlesztés a TouchGFX-re: Model-View-Presenter Design Pattern BackEnd-el

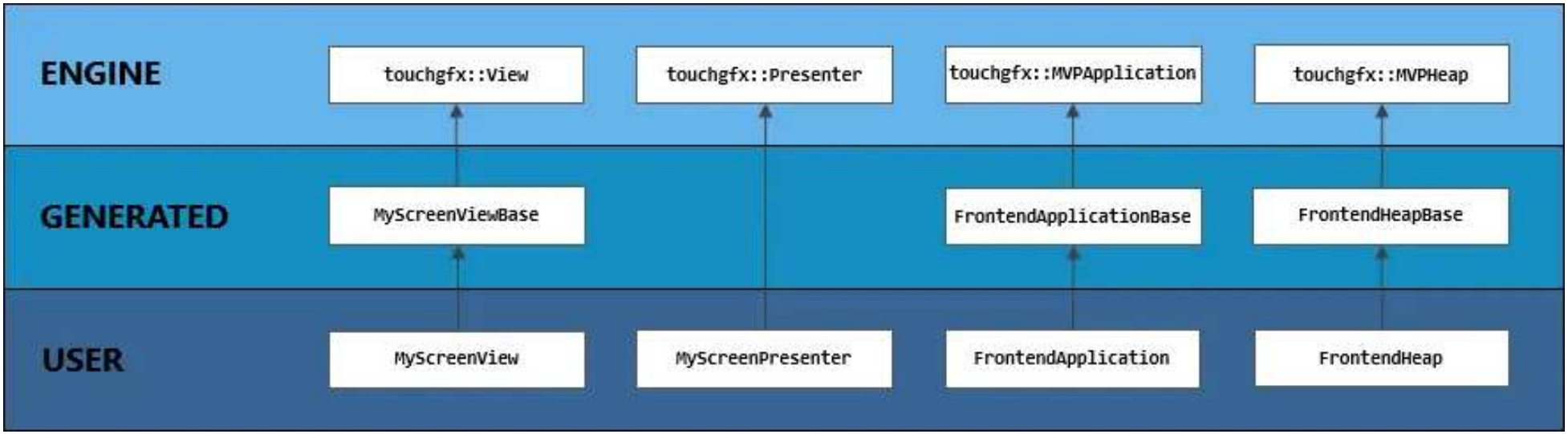


- **A Backend:** az időt igénylő hardware függő működés végrehajtása. Általában egy RTOS taszkban.

A Screen koncepció

- A TouchGFX alkalmazásban bárhány megjelenésünk, "Screens„-ünk lehet.
- A "Screens„ fogja össze a TouchGFX ben a widgets-et halmazát és azok működési logikáját.
- A „Screen” két class-ból áll: a View class: ami tartalmazza az összes screen-hez tartozó widget-et, és a Presenter class ami a megjelenítési logikát tartalmazza.
- A felhasználó eldöntheti, hogy egy vagy több „Screen”-t alkalmaz, de bonyolultabb esetben érdemes több screent alkalmazni, mert ezek szeparáltan kezelhetők

A TouchGFX designer által generált kód



- **Engine:** a TouchGFX standad class-ai. Ezek a generált class-ok alapjait szolgáltatják
- **Generated:** ezek az osztályok és a megfelelő fájlok újragenerálódnak a design változtatásakor, ezért ezeket az osztályokat és fájlokat nem szabad kézzel szerkeszteni, mivel a kézi módosítások a kódgenerátor következő futtatásakor felülíródnak. Ezek az osztályok a felhasználói osztályok alaposztályai.
- **User:** ezeket a class-okat módosíthatja a felhasználó. Ezeket nem írja felül a TouchGFX, csak legenerálja a vázukat, ha még nem léteznek.