STemWin

life.augmented

CONTENTS

- Objectives of the STemWin solution
- Basic 2D library
- PC SW Tools
- Window manager
- Configuration
- Other selected feautres

OBJECTIVES

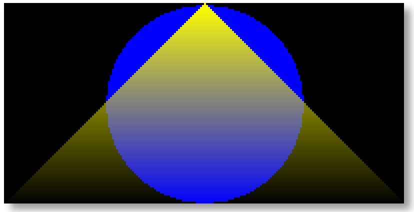
- Provide a high level graphical library which:
  - Is easy to use
  - Is flexible to customize
  - Provides graphical objects from simple 2D elements to complex window driven objects
  - Uses the ST LTDC and Chrome-ART HW features without their detailed knowledge
  - Is the industry standard in embedded field
  - Suits to project of every size → STemWin is free of charge!

CONTENTS

- Objectives of the STemWin solution
- Basic 2D library
- PC SW Tools
- Window manager
- Configuration
- Other selected feautres

Basic 2D library

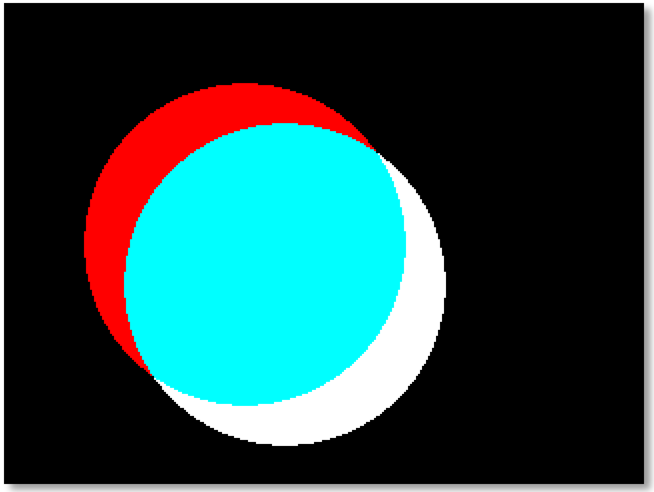
- With STemWin you can easily draw basic vector objects as
  - Lines, rectangles, arcs, polygons, graphs, bitmaps, JPEGs, PNGs and more
- Objects are drawn by a foreground color
  - `GUI_SetColor(GUI_RED);`
- STemWin supports Alpha blending – objects can overlay each other.



- Text and values can be easily written on the screen

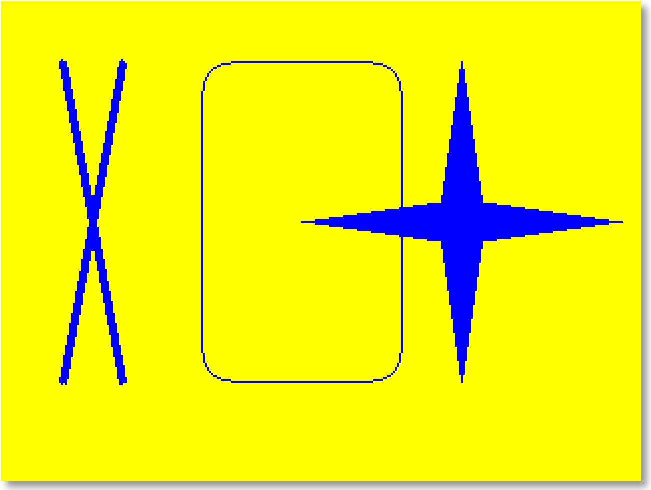
Basic 2D library - examples

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



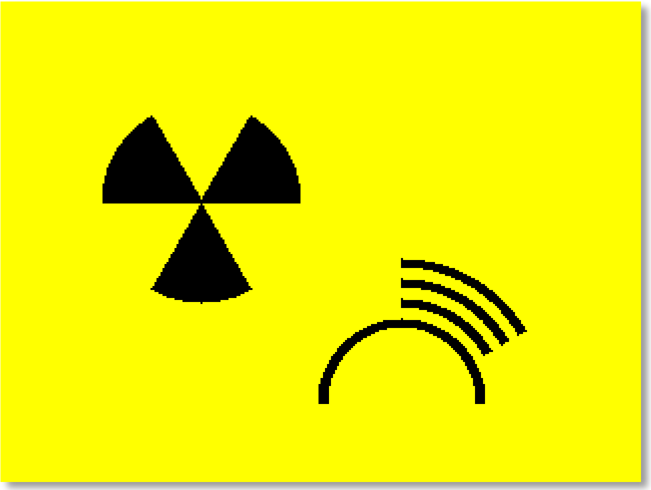
Basic 2D library - examples

```
GUI_SetBkColor(GUI_YELLOW);
GUI_Clear();
GUI_SetColor(GUI_BLUE);
GUI_SetPenSize(4);
GUI_DrawLine(30, 30, 60, 190);
GUI_DrawLine(30, 190, 60, 30);

GUI_DrawRoundedRect(100, 30, 200, 190, 15);

{
    const GUI_POINT aPoints[8] = {
    { 230, 30},{ 240, 100},
    { 310, 110}, { 240, 120},
    { 230, 190}, { 220, 120},
    { 150, 110}, { 220, 100},
    };
    GUI_FillPolygon(&aPoints, 8, 0, 0);
}
```



Basic 2D library - examples

```
GUI_DrawPie(100, 100, 50, 0, 60, 0);
GUI_DrawPie(100, 100, 50, 120, 180, 0);
GUI_DrawPie(100, 100, 50, 240, 300, 0);

GUI_DrawArc(200, 200, 40, 0, 0, 180);
GUI_DrawArc(200, 200, 50, 0, 30, 90);
GUI_DrawArc(200, 200, 60, 0, 30, 90);
GUI_DrawArc(200, 200, 70, 0, 30, 90);
```

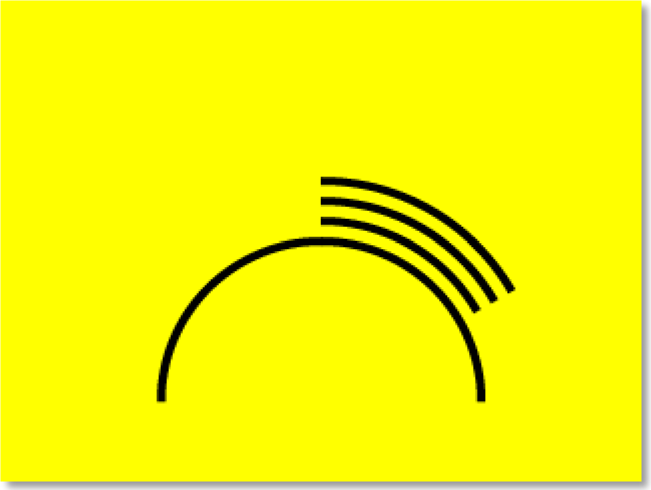


Basic 2D library - Antialiasing

- Antialiasing (AA) smoothens curves and diagonal lines by "blending" the background color with that of the foreground.
  - Text
    - Font converter is required for creating AA fonts.
  - Circles
  - Arcs
  - Lines
  - Polygons

```
GUI_AA_SetFactor(4);
GUI_AA_DrawArc(160, 200, 80, 0, 0, 180);
GUI_AA_DrawArc(160, 200, 90, 0, 30, 90);
GUI_AA_DrawArc(160, 200, 100, 0, 30, 90);
GUI_AA_DrawArc(160, 200, 110, 0, 30, 90);
```

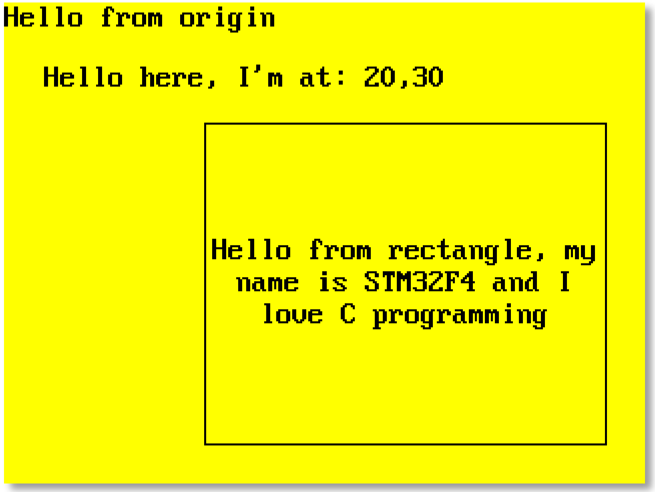
The higher the number of shades used between background and foreground colors, the better the antialiasing result (and the longer the computation time).



Basic 2D library – text

- STemWin enables you to add text of any font into your GUI
- Several API functions are available to ease text use
  - Display text at specific position
  - Manage text inside a rectangle

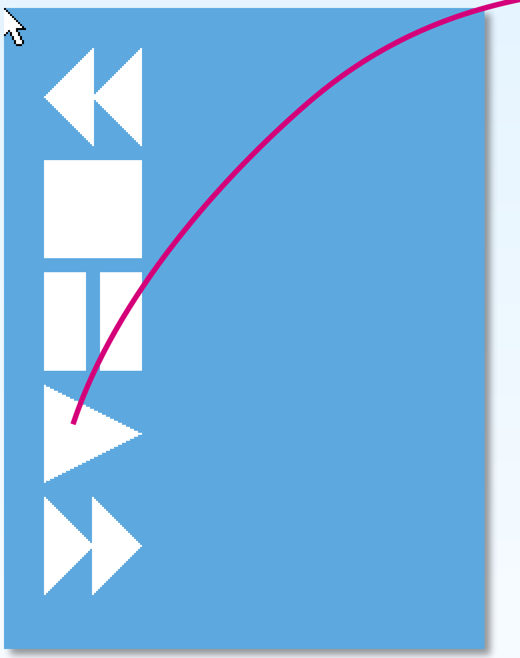
```
GUI_SetFont(&GUI_Font8x16);
GUI_DispString("Hello from origin");
GUI_DispStringAt("Hello here, I'm at:
20,30", 20,30);
{
  GUI_RECT pRect = {100, 60, 300, 220};
  GUI_DrawRect(100, 60, 300, 220);
  GUI_DispStringInRectWrap("Hello from
  rectangle, my name is STM32F4 and I love
  C programming", &pRect, GUI_TA_VCENTER |
  GUI_TA_HCENTER, GUI_WRAPMODE_WORD);
}
```



```
Hello from origin

  Hello here, I'm at: 20,30

Hello from rectangle, my
name is STM32F4 and I
love C programming
```

Example PLAYER application

- Let’s use the gained knowledge to make a simple application GUI
- With basic 2D object we can start to draw vector icons



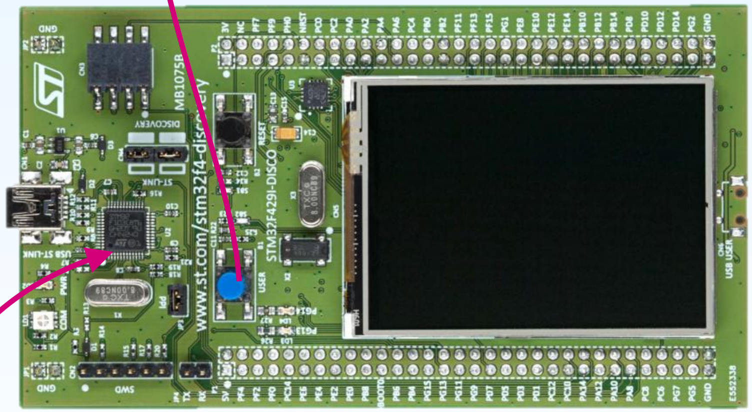
```
void DrawPlay(int x, int y, int size)
{
    GUI_POINT pPoint[3];
    pPoint[0].x = 0;
    pPoint[0].y = 0;
    pPoint[1].x = size;
    pPoint[1].y = size / 2;
    pPoint[2].x = 0;
    pPoint[2].y = size;

    GUI_FillPolygon(pPoint, 3, x, y);
}
```

- Other icons can be easily drawn very similar to PLAY icon

Example PLAYER application

- You can see the application also on the discovery kit, pushing the user button you can navigate to next steps of this example application.

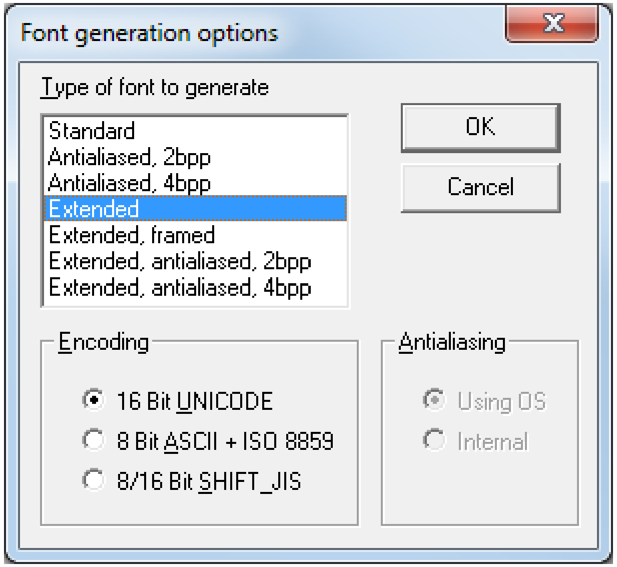
- Move forward by pushing the USER button



- Embedded STLINK debugger

Basic 2D library – Font convertor SW

- You can manage your fonts in application
  - Create fonts from any windows font
  - Manage the number of characters so that you save only those you need → save ROM
  - Export as .c files



Font generation options

Type of font to generate

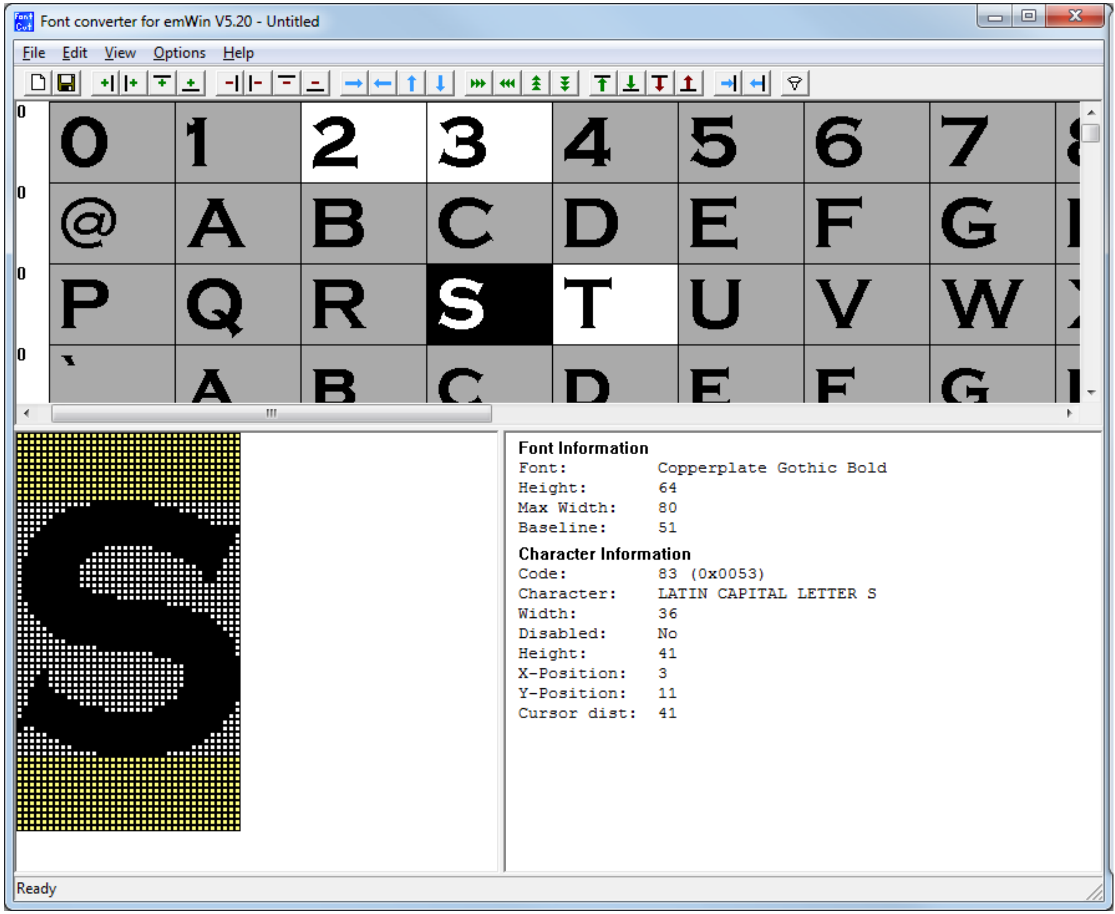
- Standard
- Antialiased, 2bpp
- Antialiased, 4bpp
- Extended
- Extended, framed
- Extended, antialiased, 2bpp
- Extended, antialiased, 4bpp

Encoding

- ● 16 Bit UNICODE
- ○ 8 Bit ASCII + ISO 8859
- ○ 8/16 Bit SHIFT_JIS

Antialiasing

- ● Using OS
- ○ Internal



Font converter for emWin V5.20 - Untitled

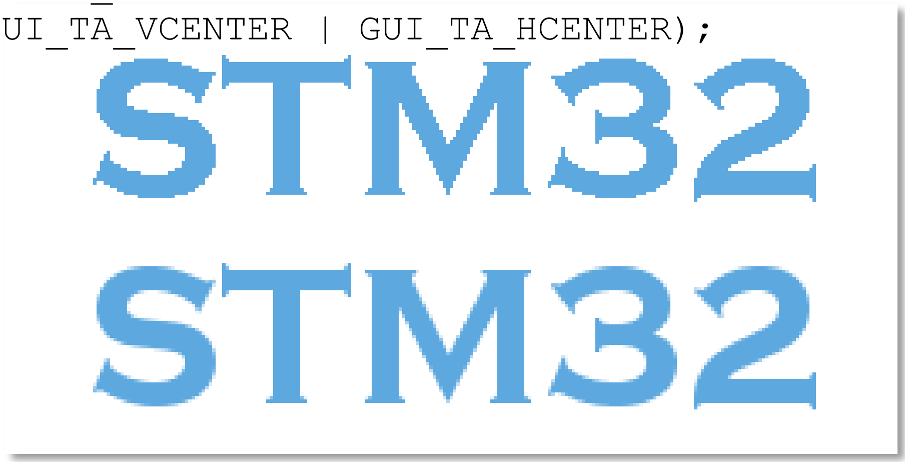
```
Font Information
Font:          Copperplate Gothic Bold
Height:        64
Max Width:     80
Baseline:      51
Character Information
Code:          83 (0x0053)
Character:     LATIN CAPITAL LETTER S
Width:         36
Disabled:      No
Height:        41
X-Position:    3
Y-Position:    11
Cursor dist:   41
```

Basic 2D library – using generated font

- Using the font we have generated is very easy
  - Include the generated .c file into the project
  - Include the extern font declaration to all modules which will use it
  - Use GUI_SetFont() function to point STemWin to this font

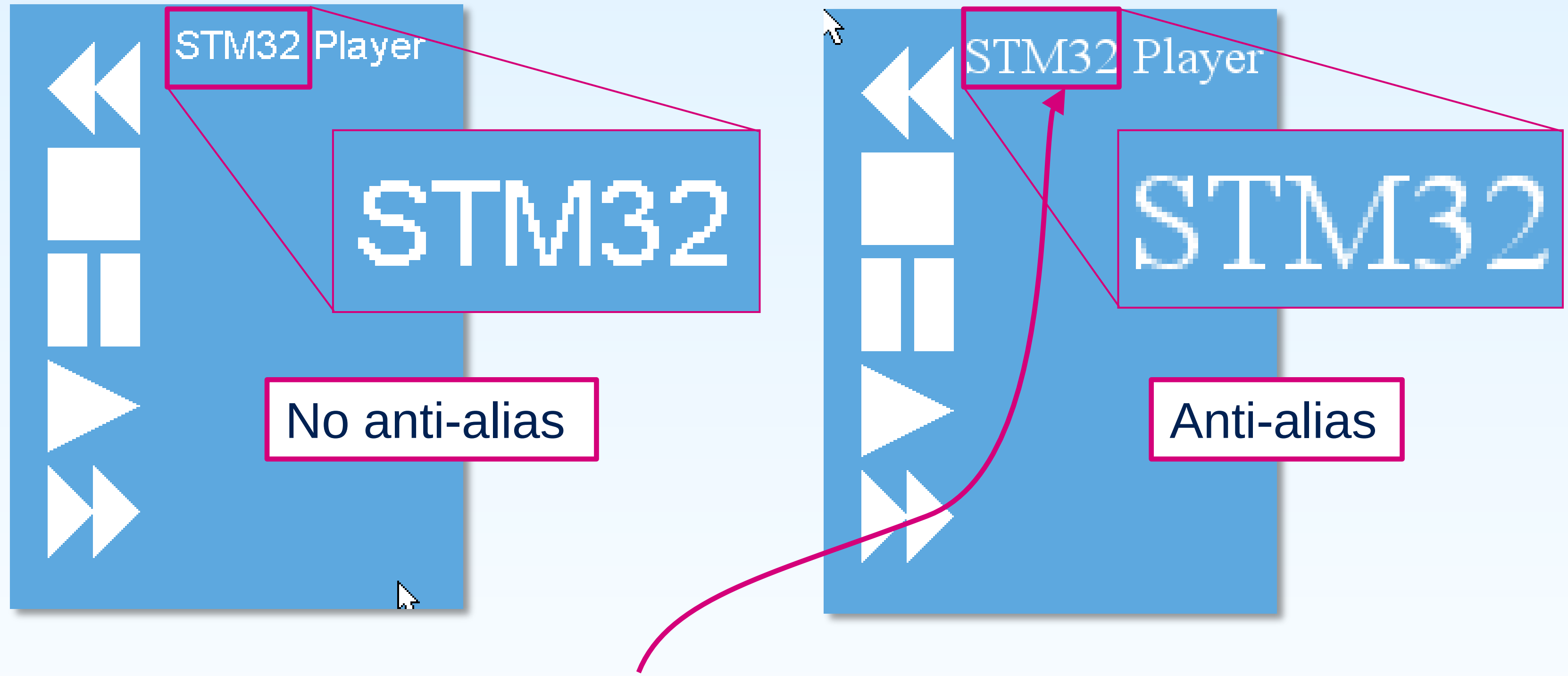
```
extern GUI_CONST_STORAGE GUI_FONT GUI_FontCopperplateGothicBold64;
extern GUI_CONST_STORAGE GUI_FONT GUI_FontCopperplateGothicBold64_aa;
..
GUI_RECT pRect1 = {0, 60, 319, 119};
GUI_RECT pRect2 = {0, 120, 319, 180};
GUI_SetFont(&GUI_FontCopperplateGothicBold64);
GUI_DispStringInRect("STM32", &pRect1, GUI_TA_VCENTER | GUI_TA_HCENTER);
GUI_SetFont(&GUI_FontCopperplateGothicBold64_aa);
GUI_DispStringInRect("STM32", &pRect2, GUI_TA_VCENTER | GUI_TA_HCENTER);
```

Drawing anti-aliased text takes much more time! (and memory as well)


STM32
STM32

Example PLAYER application

- Now we can add some text to our application



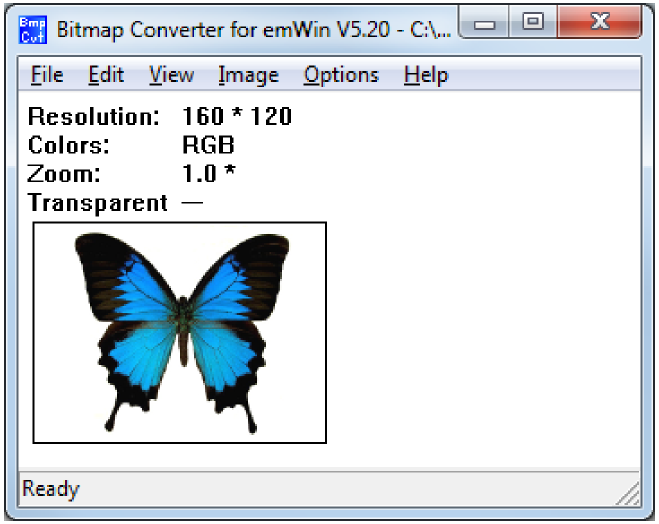
No anti-alias

Anti-alias

```
extern GUI_CONST_STORAGE GUI_FONT GUI_FontTimesNewRoman31;
GUI_SetFont(&GUI_FontTimesNewRoman31);
GUI_DispStringInRect("STM32 Player", &rect, GUI_TA_CENTER);
```

Basic 2D library bitmaps

- Static bitmaps or streamed bitmaps are supported
  - Static bitmap is completely available during drawing (e.g. stored in Flash memory)
  - Streamed bitmaps are available only by parts received by the MCU (e.g. reception from data card).
- Supported formats
  - Internal bitmap format
  - BMP
  - JPEG
  - PNG
  - GIF

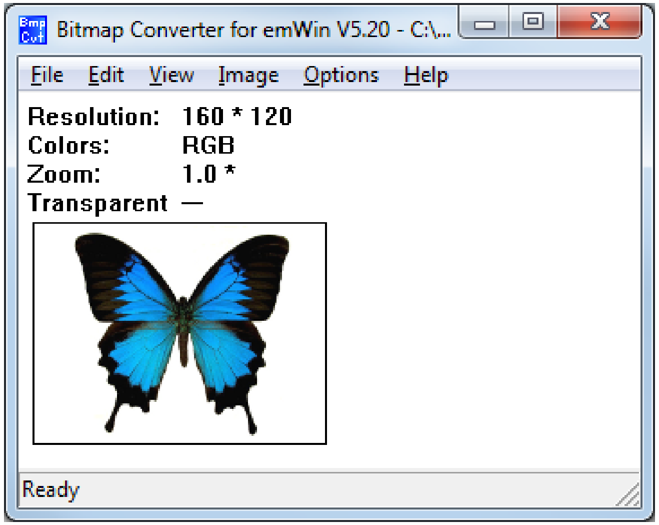


Resolution: 160 * 120
Colors: RGB
Zoom: 1.0 *
Transparent —

Ready

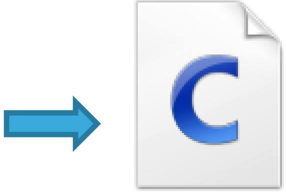
- BmpCvt.exe can be used for bitmap conversion and storage to .c files
- Bin2c.exe can be used to store any binary in .c form, e.g. complete bitmap files

Basic 2D library – bitmaps example



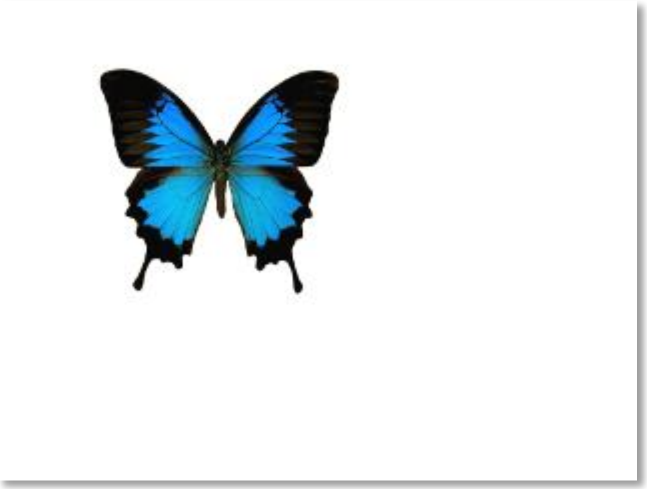
Resolution: 160 * 120
Colors: RGB
Zoom: 1.0 *
Transparent —

Ready



→ Project

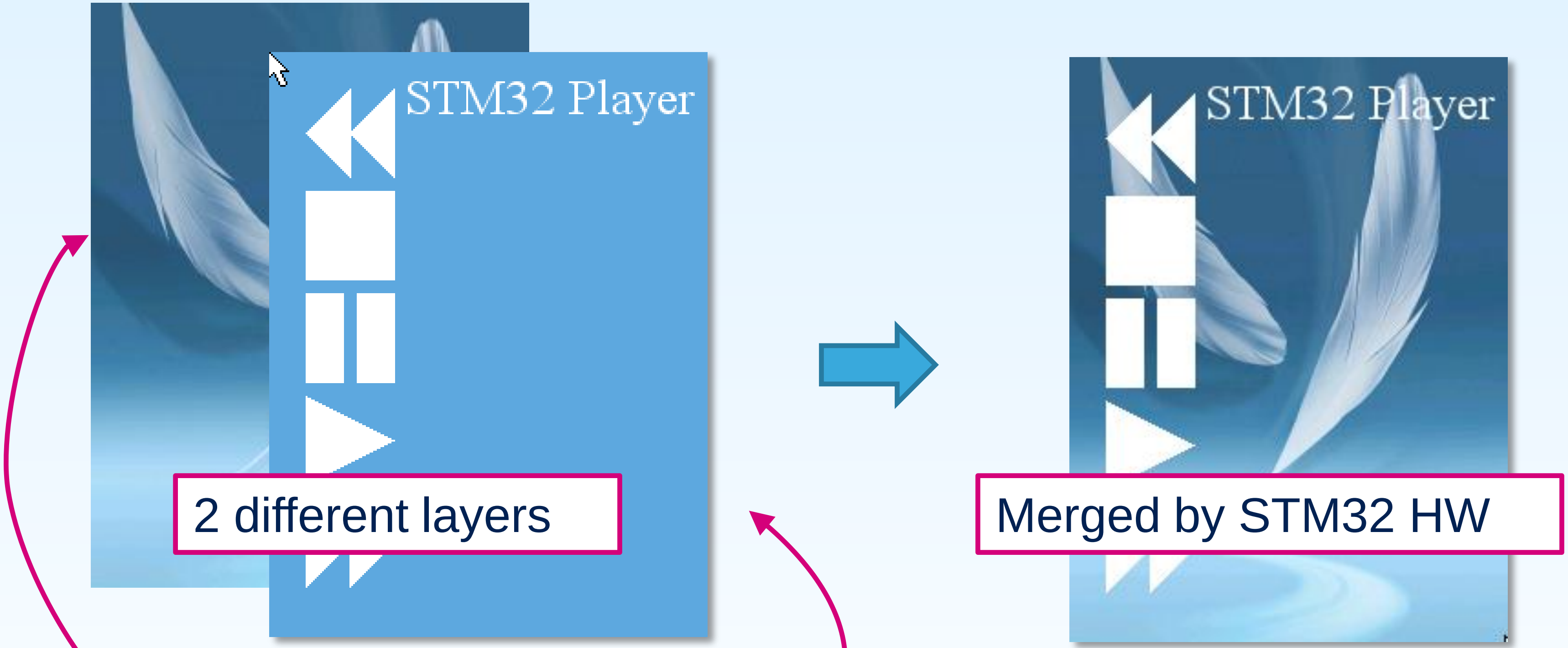
```
extern GUI_CONST_STORAGE GUI_BITMAP bmbutterfly;
GUI_DrawBitmap(&bmbutterfly, 30, 30);
```



Storing bitmap in the same format as the LCD increases drawing performance → no pixel conversion needed. But may need more storage space.

Example PLAYER application

- What about some nice background behind our player?



2 different layers

Merged by STM32 HW

```
extern GUI_CONST_STORAGE GUI_BITMAP bmbluegirl565;
GUI_SelectLayer(0); // select the background layer
GUI_DrawBitmap(&bmbackground, 0, 0);
GUI_SelectLayer(1);
// continue with drawing foreground
```

Human interface

Communication between the application and the user is mostly done by keyboard and/or **P**ointer **I**nput **D**evices or touch devices. The following functions are available:

- **GUI_StoreKeyMsg()**
  - If a keyboard event occurs (pressing or releasing a key) it should be passed to this routine.

- **GUI_PID_StoreState()**
  - If a PID event occurs (pressed, released or changing the position) it should be passed to this routine.

- **GUI_TOUCH_StoreState()**
  - In case of human touch on touchscreen, this function should be called

The WM then automatically polls the keyboard and the PID buffer. Keyboard input will be passed to the currently focused window and PID input to the respective window

Windows Manager

What is the **W**indow **M**anager?

- **Management system for a hierarchic window structure**

  Each layer has its own desktop window. Each desktop window can have its own hierarchic tree of child windows.

- **Callback mechanism based system**

  Communication is based on an event driven callback mechanism. **All** drawing operations should be done within the WM_PAINT event.

- **Foundation of widget library**

  All widgets are based on the functions of the WM.

- **Basic capabilities:**
  - **Automatic clipping**
  - **Automatic use of multiple buffers**
  - **Automatic use of memory devices**
  - **Automatic use of display driver cache**
  - **Motion support**

WM – callback mechanism

The callback mechanism requires a callback routine for each window. These routines have to support the following:

- **Painting the window**
  - Each window has to draw itself. This should be done when receiving a **WM_PAINT** message.
- **Default message handling**
  - Plain windows need to call the function `WM_DefaultProc()` to avoid undefined behavior of the window.

Further the WM needs to 'stay alive'. This can be done within a simple loop after creating the windows. It has nothing to do but calling `GUI_Delay()` which does the following:

- **PID management**
- **Key input management**
- **Timer management**

WM – callback mechanism

The callback mechanism requires a callback routine for each window. These routines have to support the following:

- **Painting the the window**
    - Each window has to draw itself. This should be done when receiving a **WM_PAINT** message.
- **Default message handling**
    - Plain windows need to call the function `WM_DefaultProc()` to avoid undefined behavior of the window.

Further the WM needs to 'stay alive'. This can be done within a simple loop after creating the windows. It has nothing to do but calling `GUI_Delay()` or `GUI_Exec()` which does the following:

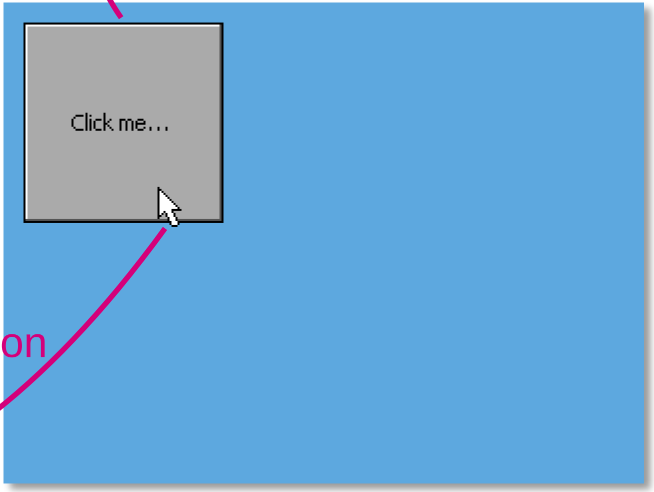
- **PID management**
- **Key input management**
- **Timer management**

```
hButton = BUTTON_Create( 10, 10, 100, 100, GUI_ID_BUTTON0, WM_CF_SHOW);
BUTTON_SetText(hButton, "Click me...");
WM_SetCallback(WM_HBKWIN, myCbBackgroundWin);
WM_SetCallback(hButton, myCbButton);
... ... ...
static void myCbBackgroundWin(WM_MESSAGE *pMsg) {
  int NCode, Id;
  switch (pMsg->MsgId) {
  case WM_NOTIFY_PARENT:
    Id    = WM_GetId(pMsg->hWinSrc);      /* Id of widget */
    NCode = pMsg->Data.v;                 /* Notification code */
    if ((Id == GUI_ID_BUTTON0) && (NCode == WM_NOTIFICATION_RELEASED))
      buttonClicked = 0;
   break;
   case WM_PAINT:
      GUI_SetBkColor(STBLUE);
      GUI_Clear();
   break;
  }
}
... ... ...
static void myCbButton(WM_MESSAGE *pMsg) {
   switch (pMsg->MsgId) {
   case WM_TOUCH:
   if (((GUI_PID_STATE*)(pMsg->Data.p))->Pressed == 1)
      buttonClicked = 1;
      break;
   case WM_SIZE:
   // add some code
   break;
   default:
      BUTTON_Callback(pMsg);
   }
}
```

Initialization – called only once!

Redraw part

On release WM calls parent callback function with message informing about child touch release



On click WM calls Button callback function

Default callback must be called to achieve proper functionality

WM – Widget library

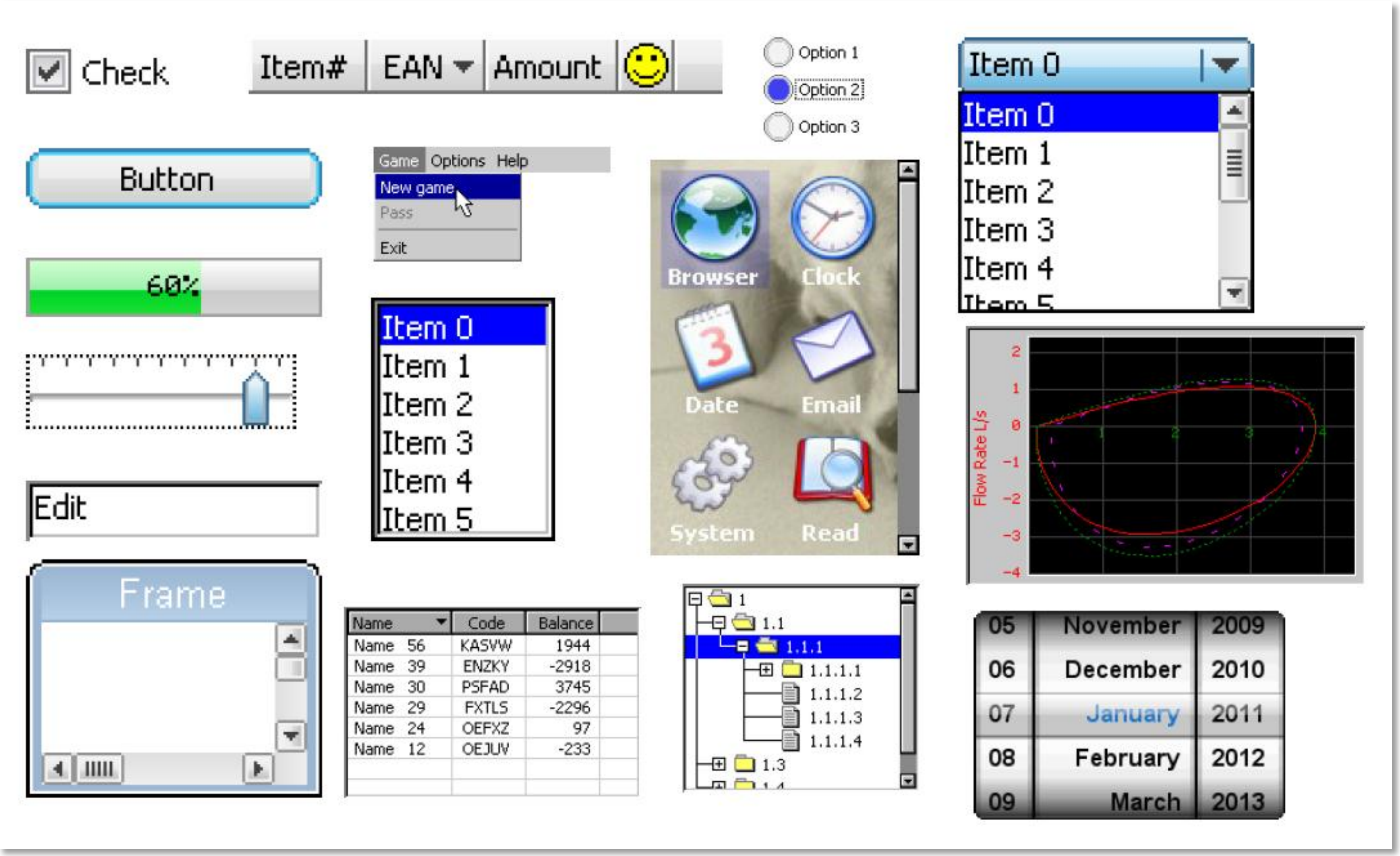
Widget = **Wi**ndow + Ga**dget**

Currently the following widgets are supported:

- Button, Checkbox, Dropdown, Edit, Framewin, Graph, Header, Iconview, Image, Listbox, Listview, Listwheel, Menu, Multiedit, Progbar, Radio, Scrollbar, Slider, Text, Treeview

- Creating a widget can be done with one line of code.
- There are basically 2 ways of creating a widget:
  - **Direct creation**
    For each widget there exist creation functions:
    <WIDGET>_CreateEx() // Creation without user data.
    <WIDGET>_CreateUser() // Creation with user data.
  - **Indirect creation**
    A widget only needs to be created indirectly if it is to be included in a dialog box.
    <WIDGET>_CreateIndirect() // Creates a widget to be used in dialog boxes.

WM – Widget library



Name	Code	Balance
Name 56	KASVW	1944
Name 39	ENZKY	-2918
Name 30	PSFAD	3745
Name 29	FXTLS	-2296
Name 24	OEFXZ	97
Name 12	OEJUV	-233

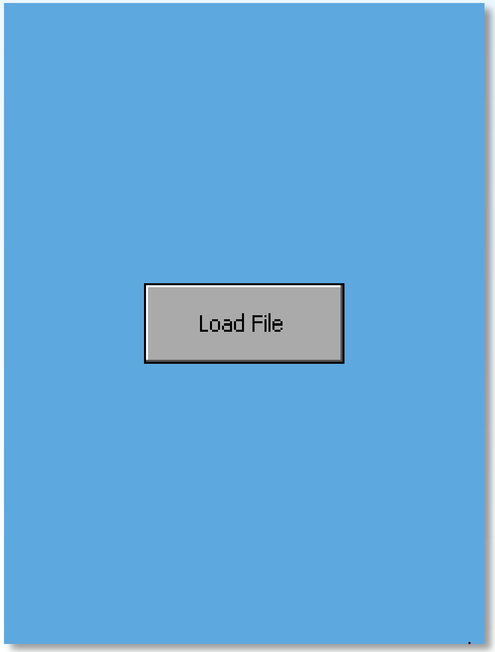
Example PLAYER application

- Let’s rework our PLAYER example using WM.
- Start with single button for prompt the user to select song

```
#include "BUTTON.h"
static BUTTON_Handle hButton;

hButton = BUTTON_Create( 320/2 - BUTTON_WIDTH / 2, 240/2 -
BUTTON_HEIGHT / 2, BUTTON_WIDTH, BUTTON_HEIGHT, GUI_ID_OK,
WM_CF_SHOW);
BUTTON_SetText(hButton, "Load File");

GUI_Exec();
```

Always include the widget header

Only after call of this function the windows get paint

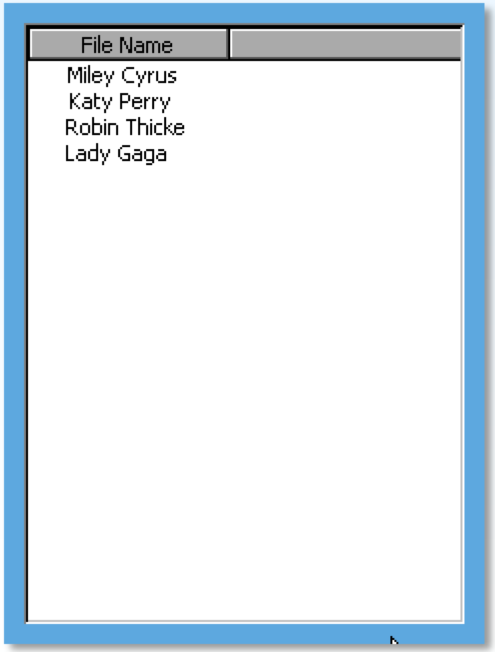


Example PLAYER application

- After click on button show a LISTVIEW widget with selection of songs

```
hListView = LISTVIEW_Create(320/2 - LISTVIEW_WIDTH / 2, 240/2 -
LISTVIEW_HEIGHT / 2, LISTVIEW_WIDTH, LISTVIEW_HEIGHT, WM_HBKWIN,
GUI_ID_LISTVIEW0, 0, 0);

LISTVIEW_AddColumn(hListView, 100, "File Name", GUI_TA_CENTER);
for (i = 0; i < GUI_COUNTOF(myFileTable); i++) {
   LISTVIEW_AddRow(hListView, myFileTable[i]);
}
```

In this loop just copy text array into the widget



File Name	
Miley Cyrus	
Katy Perry	
Robin Thicke	
Lady Gaga	

Example PLAYER application

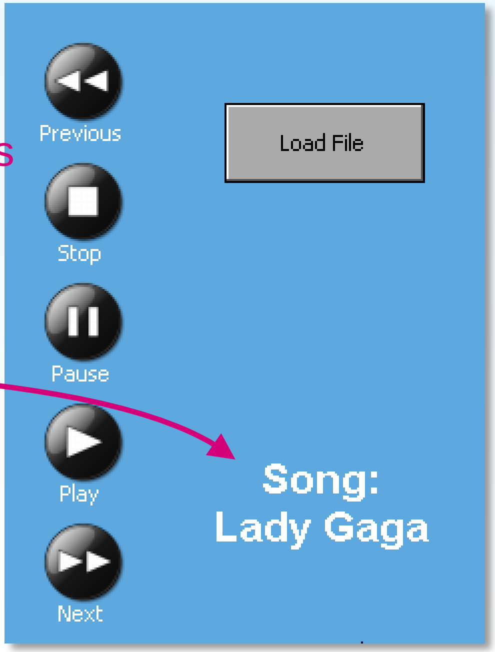
- Now we can use ICONVIEW widget to change vector icons by nice semi-transparent bitmap icons
- Simply use icons in .png, convert them to .c by bitmap converter, add to project and use as ICONVIEW widget source:

```
hIconView = ICONVIEW_CreateEx(ICONS_HOR_POS, ICONS_VER_POS, LCD_GetXSize() -
2*ICONS_HOR_POS, bmStyleNexticon1.YSize + GUI_Font8_1.YSize + 10, WM_HBKWIN,
WM_CF_SHOW, ICONVIEW_CF_AUTOSCROLLBAR_V, GUI_ID_ICONVIEW0,
bmStyleNexticon1.XSize, bmStyleNexticon1.YSize + GUI_Font8_1.YSize);
...
for (i = 0; i < iconNumber; i++) {
    ICONVIEW_AddBitmapItem(hIconView, myBitmapItem[i].pBitmap,
myBitmapItem[i].pText);
}
```

In this loop just copy pointers to icon bitmaps and texts into the widget

Text may appear also as widget TEXT

```
hTextSong = TEXT_CreateEx (0, 200, 320 - 1, 30,
WM_HBKWIN, WM_CF_SHOW, 0, GUI_ID_TEXT0,
pSongString);
```

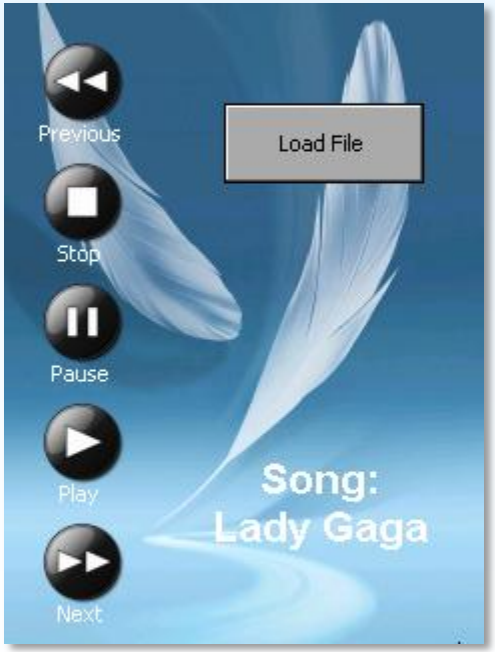


Example PLAYER application

- Finally put the bitmap background into background layer
- To do so, we only need to switch to LCD background layer
  - In reality this only changes the frame buffer start address for drawing operations→ no performance lost
- Then draw bitmap as done before

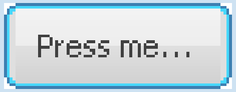
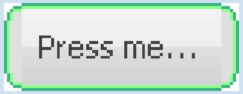
```
GUI_SelectLayer(0); // select background layer
GUI_DrawBitmap(&bmbackground, 0, 0);
GUI_SelectLayer(1);
```

Select back the foreground layer for all subsequent operations



WM – Skinning

- Skinning is a method of changing the appearance of one or multiple widgets.
- A skin is just a simple callback function which is available for drawing all details of a widget.
- These widgets can be skinned:
  - BUTTON
  - CHECKBOX
  - DROPDOWN
  - FRAMEWIN
  - HEADER
  - PROGBAR
  - RADIO
  - SCROLLBAR
  - SLIDER

Before	After
	

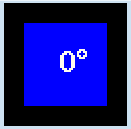
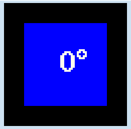
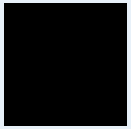
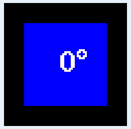
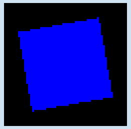
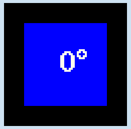
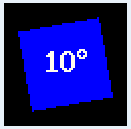
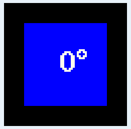
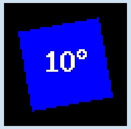
Interfacing to HW, configuration

- STemWin is high level GUI package, but it needs a low level driver to access MCU resources
- In the package, there are 2 drivers:
  - STM32F4 LTDC customized driver which benefits from Chrome-ART acceleration for copying/fill routines
  - FlexColor driver used for interfacing external LCD drivers through FSMC interface. E.g. popular Ilitek ILI9320, ILI9325 or Himax HX8347 devices.
- Configuration through LCD_Conf.c, for example:

```
#define GUI_NUM_LAYERS 2   // defines how many layers are used
#define XSIZE_PHYS 480     // set-up the physical dimensions
#define YSIZE_PHYS 272
// and the color mode for layer 0
#define COLOR_MODE_0 CM_ARGB8888
// physical address of the frame buffer
#define LCD_FRAME_BUFFER ((U32)0xC0000000)
```

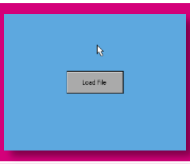
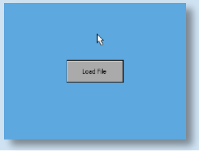
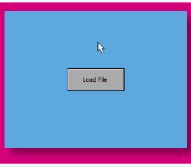
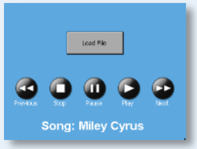
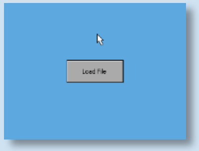
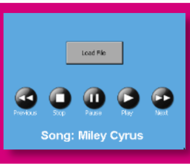
Other STemWin features

- Memory devices
  - Memory Devices can be used in a variety of situations, mainly to **prevent the display from flickering** when using drawing operations for overlapping items.
  - This requires additional RAM memory

API function	Without Memory Device	With Memory Device
Step 0: Initial state		
Step 1: GUI_Clear()		
Step 2: GUI_DrawPolygon()		
Step 3: GUI_DispString()		
Step 4: GUI_MEMDEV_CopyToLCD()		

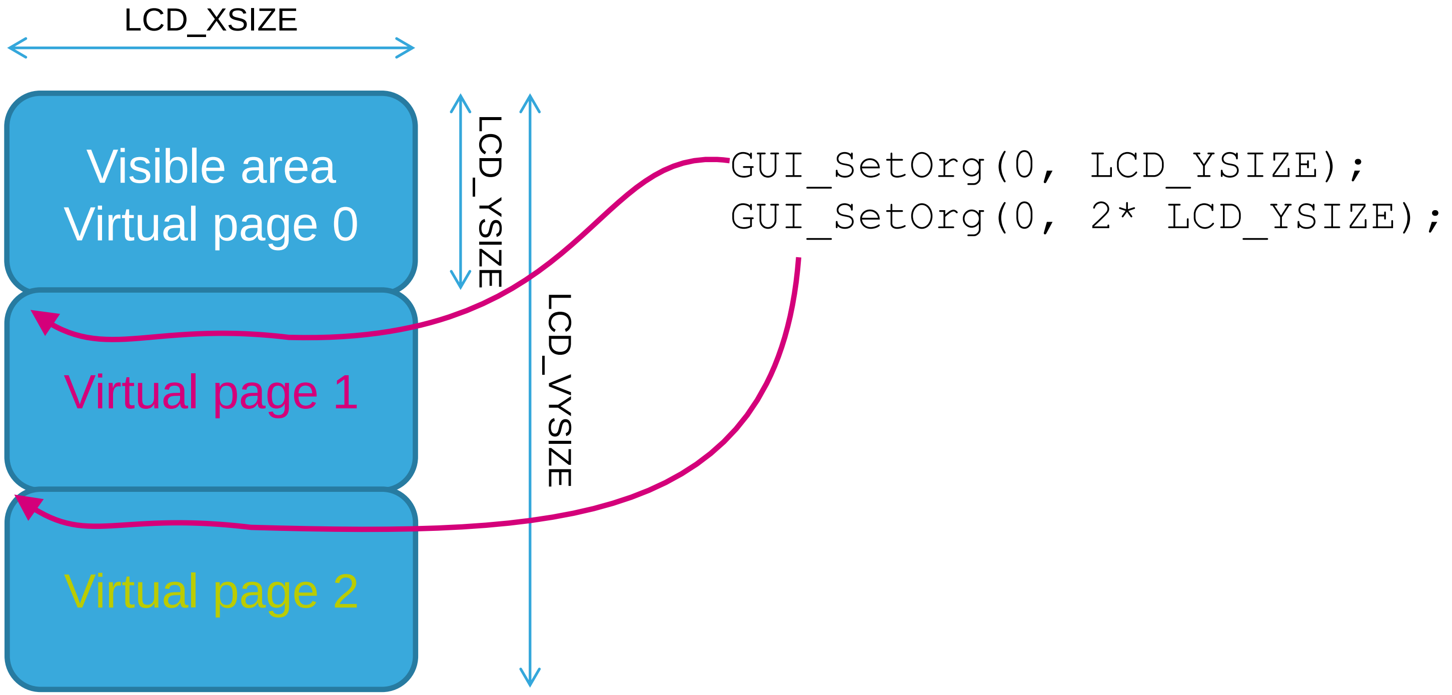
Other STemWin features

- Multiple buffering
  - With multiple buffers enabled there is a front buffer which is used by the display controller to generate the picture on the screen and one or more back buffers which are used for the drawing operations.
- In general it is a method which is able to avoid several unwanted effects:
  - The visible process of drawing a screen item by item
  - Flickering effects caused by overlapping drawing operations
  - Tearing effects caused by writing operations outside the vertical blanking period

API function	1st buffer	2nd buffer
1. Copy active buffer to inactive		
1. Drawing operations in inactive buffer		 Load File Previous Stop Pause Play Next Song: Miley Cyrus
3. Set 2nd buffer as active		 Load File Previous Stop Pause Play Next Song: Miley Cyrus

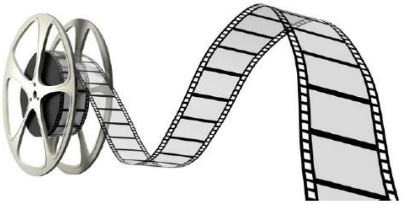
Other STemWin features

- Virtual screens
- Display area greater than the physical size of the display. It requires additional video memory and allows instantaneous switching between different screens.
- Similar to Multiple buffering, but no copy operation is performed.

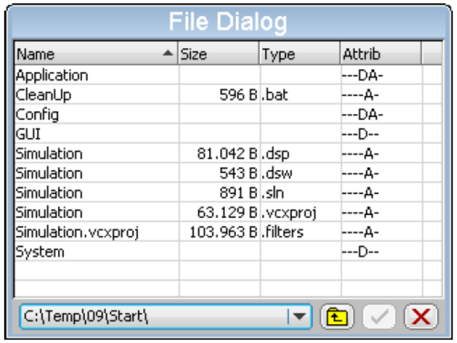


```
GUI_SetOrg(0, LCD_YSIZE);
GUI_SetOrg(0, 2* LCD_YSIZE);
```

Other STemWin features



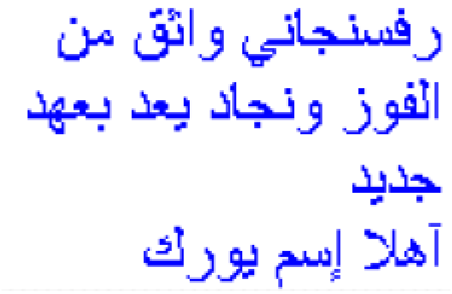
- And many others:
  - Motion JPEG video
  - Support for real time operating systems
  - Dialogs
  - GUIBuilder SW
  - Sprites
  - Arabic, Thai, Chinese, Japanese fonts
  - VNC Server



File Dialog

Name	Size	Type	Attrib
Application			---DA-
CleanUp	596 B	.bat	----A-
Config			---DA-
GUI			---D--
Simulation	81.042 B	.dsp	----A-
Simulation	543 B	.dsw	----A-
Simulation	891 B	.sln	----A-
Simulation	63.129 B	.vcxproj	----A-
Simulation.vcxproj	103.963 B	.filters	----A-
System			---D--

C:\Temp\09\Start\



رفسنجاني واثق من
الفوز ونجاد يعد بعهد
جديد
آهلا إسم يورك

www.st.com/stm32f4