HiGV

FAQs

Issue 00B03

Date 2018-12-29

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: http://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes solutions to the problems that engineers may encounter when using the HiGV system during development.

NOTE

This document takes Hi3559A V100 as an example. Unless otherwise specified, the contents of Hi3559A V100 also apply to other chips.

Related Version

The following table lists the product versions related to this document.

Product Name	Version
Hi3559A	V100
Hi3559C	V100
Hi3556A	V100
Hi3519A	V100
Hi3559	V200
Hi3556	V200
Hi3518E	V300
Hi3516D	V300
Hi3516C	V500

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 00B03 (2018-12-29)

This issue is the third draft release, which incorporates the following changes:

The contents related to the Hi3556A V100/Hi3519A V100/Hi3559 V200/Hi3556 V200 are added.

Issue 00B02 (2018-02-26)

This issue is the second draft release, which incorporates the following changes:

Section 1.1.15 and section 1.1.16 are modified.

Issue 00B01 (2017-08-26)

This issue is the first draft release.

Contents

Figures

1 FAQs

1.1 HiGV FAQs

1.1.1 How do I Set the Control Skin Transparency?

[Description]

Some graphical user interface (GUI) scenarios require translucent background skin.

[Analysis]

You can set the skin by using a translucent image or setting the color attribute to translucent.

[Solution]

The skin is set in the **.xml** skin description file. Two skin types are supported: **pic** and **color**.

The methods of setting the skin to translucent are as follows:

- To set a translucent image, process the background image with image-editing software, change the transparency to translucent, and then reference the processed image in the **.xml** skin file.
- Set the color attribute to translucent. For example, the color attribute format of 0xFF000000 indicates Alpha, Red, Green and Blue (ARGB). The two bytes FF in the most significant bit (MSB) indicates the Alpha transparency which ranges from 00 (transparent) to FF (opaque). Set the two bytes to any value between 00 and FF to achieve the translucent effect.

1.1.2 How Do I Set the Dot-matrix and Vector Font Libraries?

[Description]

Fonts are used frequently on GUIs. HiGV supports settings of dot-matrix and vector font libraries.

[Analysis]

Compared with the vector font library, the dot-matrix font library occupies less flash memory but cannot be scaled and has less rounder font edge. To achieve better display effect, the vector font library is recommended.

[Solution]

- Font libraries are added in the .xml font file. The following shows how to add a dot-matrix font library and vector font library, respectively.

  Setting the dot-matrix font library:

```
<font
    id="Font10"
    sbname="./res/font/ubf/Font10.ubf"
    mbname=""
    size=""
    isbold=""
isitalic=""/>
```

  Setting the vector font library:

```
<font
    id="Font40"
    sbname="./res/font/ttf/basic.ttf"
    mbname=""
    size="40"
    isbold=""
isitalic=""/>
```

- The obvious difference between settings of dot-matrix fonts and vector fonts is that the latter requires size setting but the former size is fixed and does not require size setting.
- The extension of the dot-matrix font library is **.ubf**, while the extension of the vector font library is **.ttf**.
- You can run the HiFontTool tool to convert a vector font library into a dot-matrix font library.
- The current **.xml** font file supports settings of eight fonts at most. If you set a value beyond the range, it does not take effect.

1.1.3 How Do I Dynamically Set the Pixel Format of the Graphics Layer for the Window

[Description]

This topic describes how to set the pixel format of the graphics layer at which the window locates when the window is switched, for example, changing from HIGO_PF_4444 is HIGO_PF_1555. A typical application scenario is to support an application to ensure good effect of graphics layer rotation, translucent display, and thumbnail display at the same time.

[Analysis]

HiGO provides the HI_GO_ChangeLayerFmt interface for you to set the pixel format after the window is switched.

[Solution]

If a window A is switched to window B and the pixel format is changed from HIGO_PF_4444 to HIGO_PF_1555, the procedure is as follows:

Step 1 Set the window mode.

In the **.xml** layout file of window A, add **pixelformat** to the window attribute, and set it to **argb4444**. Similarly, add **pixelformat=argb1555** to the window attribute in the **.xml** layout file of window B.

Step 2 Register the callback function.

In the **.xml** file for switching window B, register the **onshow** callback function, in which you can set the interface for switching the pixel format.

Step 3 Set the pixel format.

In the **onshow** callback function, call the HI_GV_Layer_GetHigoLayer interface first to obtain the handle to the HiGO graphics layer, and then call the HI_GO_ChangeLayerFmt interface to set the pixel format of the graphics layer at which the window locates. If the layer needs to be rotated, call the HI_GV_Layer_SetRotateMode interface again to set the graphics layer rotation.

----End

1.1.4 How Do I Batch Modify the Layout Files?

[Description]

This topic describes how to proportionally scale contents in **.xml** layout files.

[Analysis]

You can batch modify **.xml** layout files on the Linux server by using the **xml2bin** tool in the release package.

[Solution]

The following shows an example of running a batch modification command:

 # ~/../xml2bin-x xml/ -C0320024004800320-bbin/higv.bin-tc-esrc/extcfile/-lsrc/extcfile/

- This command indicates that the canvas resolution of the layout files in the **xml** directory is changed from 320 x 240 to 480 x 320.
- The command parameter description can be obtained by running the **./xml2bin-H** command.

1.1.5 How Do I Batch Scale Images?

[Description]

This topic describes how to batch scale a large number of images.

[Analysis]

You can run the commands of the open-source software FFmpeg to scale images.

[Solution]

- Source codes of FFmpeg commands need to be downloaded from the official website (http://ffmpeg.org/). Implement FFmpeg software compilation and installation on the Linux. If the width and height of an image need to be scaled to half of the original size, run the following command:

  #ffmpeg -i SING0001.JPG -vf scale=iw*0.5:ih*0.5 output.JPG
- The following shows the batch scaling command:

#find . -name "*.png" | xargs -I {} ffmpeg -y -i {} -vf scale=iw*0.75:ih*0.75 {}

iw indicates the original image width, **ih** indicates the original image height, and the scaling ratio is 0.75.

1.1.6 What Do I Do If the Font Turns Bold?

[Description]

The font gradually turns bold during the display or it is not displayed as expected.

[Analysis]

Generally, this problem is caused by repeated drawing of fonts.

[Solution]

- View the **.xml** layout file and check whether overlapping issues occur among font controls. If yes, remove overlay regions by parameter setting.
- View the code file and check whether the font drawing interface is frequently and repeatedly called. Repeated drawing of fonts is not allowed.

1.1.7 What Do I Do If the Translucency of Control Skin Is not Displayed?

[Description]

When the control skin has been set to translucent, the opaque or transparent effect is displayed on the screen.

[Analysis]

The skin transparency is affected by the pixel format of the graphics layers. Some pixel formats do not support the translucent display.

[Solution]

- Currently the Hi3559A V100 chip supports pixel formats of HIGO_PF_4444, HIGO_PF_1555, and HIGO_PF_8888, in which only HIGO_PF_4444 and HIGO_PF_8888 support setting of translucent skin. HIGO_PF_1555 does not support direct setting of translucent skin but allows extended translucency.
- If you have set the pixel format to one that supports translucent skin display, but the translucent effect fails to appear, and the skin type is **color**, try to change the skin type to **pic** and use a translucent image as the background skin.

1.1.8 How Do I Solve GUI Issues about Portrait Orientation of the Screen?

[Description]

An error is reported to the serial port during portrait orientation, and the GUI does not appear.

[Analysis]

Screen display is divided into landscape orientation and portrait orientation according to the display direction. Generally the GUI layout of motion DV and event data recorder (EDR) products is set to landscape orientation, which suits human visual habits better.

[Solution]

Step 1 Initialize the graphics layer.

To set the graphics layer parameters, reverse the width and height of the canvas as follows:

HIGO_LAYER_INFO_S LayerInfo = {SCREEN_WIDTH, SCREEN_HEIGHT, **SCREEN_HEIGHT, SCREEN_WIDTH**, SCREEN_WIDTH, SCREEN_HEIGHT, ……};

The three groups of width and height information of the graphics layer are parameters of the screen, canvas, and displayed image. Reverse the second group of parameters, namely canvas parameters, to achieve the horizontal layout. For current Hi3559A V100 chips, the pixel formats that support rotation are HIGO_PF_4444, HIGO_PF_1555, and HIGO_PF_8888.

Step 2 Call the rotation interface.

After the graphics layer initialization is complete, call the rotation interface HI_GV_Layer_SetRotateMode to rotate the graphics layer by 90°or 270°. For details about interface configuration, see the **middleware/sample/higv/scene_sample/main_sample.c** file in the release package directory.

----End

1.1.9 What Do I Do If the Message Queue Is Full?

[Description]

Intermittence and discontinuity occur during the GUI display. The message of "Send async msg failed, queue full." is displayed over the serial port.

[Analysis]

The HiGV internal drawing is implemented through a message distribution mechanism. When the message queue is blocked, drawing messages will not be distributed to the controls. As a result, the user interface (UI) drawing is not implemented, causing phenomena of intermittence and discontinuity.

[Solution]

- Message distribution is implemented in the UI main thread. If the main thread is blocked, the message queue will be blocked as well.
- Generally this issue results from time-consuming operations in the callback function that you registered on the xml control. As the callback function is in the main thread, time-consuming operations will block the entire main tread, causing phenomena of UI intermittence and discontinuity. A good approach to solving such issues is performing time-consuming operations in sub threads to avoid influencing the proper running of the main tread.

1.1.10 What Do I Do If the Thumbnail Effect Is Poor in the Playback GUI?

[Description]

When the Scroll Grid control is used for 9-picture or 6-picture thumbnail display, the thumbnails show poorer display effect and less color gradient than that of original images.

[Analysis]

The red-green-blue (RGB) bits of the pixel format of the graphics layer have a great influence on the color gradient. The lower the pixel format bit is, the poor the display effect becomes. You can use a pixel format with a large number of bits to achieve a better effect.

[Solution]

For initializing the graphics layer, if the pixel format is HIGO_PF_4444, change it to HIGO_PF_8888 or HIGO_PF_1555.

1.1.11 What Do I Do If the Animation Framework Moves Unsmoothly?

[Description]

When the HiGV animation interface is used for moving controls, for example, moving a Group Box control into the screen from the outside, intermittence and discontinuity issues occur during the movement.

[Analysis]

The movement of a control is a process of continuous drawing. Therefore, reducing the drawing time as much as possible can efficiently improve the performance of control movement.

[Solution]

- Call the interface HI_GV_Widget_Move that internally provides the drawing function for control movement, so that you do not need to call HI_GV_Widget_Update, HI_GV_Widget_Paint, or other interface for repeated drawing.
- View the interfaces called during the movement, check whether the debugging of information or other redundant operation exists, and remove them such operations.

1.1.12 What Do I Do If an Error of Concurrent Interface Calling Occurs?

[Description]

HiGV interfaces are called in multiple sub threads to implement display, hide, drawing and other operations, resulting in unexpected exceptions.

[Analysis]

To improve the efficiency of message distribution and drawing performance, most HiGV internal interfaces do not support multiple threads access protection. As a result, interfaces are called in sub threads, possibly causing unexpected issues.

[Solution]

- The HiGV is a single-thread module, and all interface calling must be implemented in the main thread. Do not call HiGV interfaces in sub threads.
- If a sub thread needs to instruct controls to perform processing, this service can be implemented through asynchronous message transmission. The asynchronous message transmission interface is HI_GV_Msg_SendAsync, which supports multiple threads access protection. The sub thread can be used but the precondition is that the message processing function corresponding to the control has been registered in the main thread. For Sample code implementation, see the **asy_msg_scene.c** file in the **middleware/sample/higv/scene_sample** release package directory.

NOTICE

For details about asynchronous message transfer interface, see the **hi_gv_msg.h** file. HI_GV_Msg_SendAsync can meet the requirements of most scenarios. If a segment of memory data needs to be sent to the main thread, call the HI_GV_Msg_SendAsyncWithData interface.

1.1.13 What Do I Do If GUI Overlaying Appears When the List Box Control is Scrolled?

[Description]

After the List Box control development is complete, slide up or down the control content on the touchscreen, drawing overlaying issue occurs on the user interface (UI).

[Analysis]

Control movement is a process of continuous UI drawing. The drawing depends on the skin resources and requires checking of the control skin which is correctly set or not.

[Solution]

This issue is generally caused by the List Box control settings in which only **Normalskin** is set but **Activeskin** is not set. You need to check the skin settings in the **.xml** layout file.

1.1.14 How Do I Solve Crack Issues Caused by Fast-moving Controls?

[Description]

In touchscreen scenarios, your finger moves back and forth on the List Box or Scroll View controls, crack issues of control contents appear with high probabilities.

[Analysis]

There are three HiFB refresh modes, namely, 0 buffer, 1 buffer, and 2 buffer, which are defined as follows:

- 0 buffer (HIFB_LAYER_BUF_NONE)

  The upper user canvas buffer is the display buffer. This refresh type can reduce memory consumption with the fastest speed, but users can view the drawing process of the image.
- 1 buffer (HIFB_LAYER_BUF_ONE)

  The display buffer is provided by the HiFB. Therefore, a certain memory is required. Considering the display effect and memory requirements, this refresh type is selected, but the picture alias occurs.
- 2 buffer

  The display buffer is provided by the HiFB. Compared with the preceding refresh types, the type requires the most memory but provides the best image display effect.

[Solution]

In conclusion, 0 buffer and 1 buffer refresh modes have certain disadvantages in display and may cause crack issues. Therefore, it is required to change the refresh mode to 2 buffer.

Correspondingly, for initializing a graphics layer in a HiGV application, set the refresh mode parameter to 2 buffer, namely, the transmitted parameter is HIGO_LAYER_BUFFER_OVER.

```
HIGO_LAYER_INFO_S LayerInfo = {……,
(HIGO_LAYER_FLUSHTYPE_E)(HIGO_LAYER_BUFFER_OVER),
HIGO_LAYER_DEFLICKER_AUTO, HIGO_PF_1555, HIGO_LAYER_HD_0};
```

1.1.15 What Do I Do If the Callback Function of the Touch Gesture Event Is Invalid During Touchscreen Adaptation?

[Description]

In touchscreen scenarios, when a finger slides on the touchscreen or touch the screen, the gesture callback function of the control does not respond.

[Analysis]

During touchscreen adaptation, the **touchpadadp.c** file must be modified. This file is used to perform the following operations:

- HI_GV_TOUCH_OpenDevice and HI_GV_TOUCH_CloseDevice

  You can enable or disable touchscreen devices by calling these two interfaces.
- HI_HAL_TOUCHPAD_Enable

  This interface is used to enable the related flag bit so that the touchscreen data can be read properly.
- HI_GV_TOUCH_ReadData

  This interface is used to read data related to touchscreen operations.

  ```
  typedef struct hiHIGV_TOUCH_INPUTINFO_S
  {
      HI_S32     id;/**<input id info, one finger or two fingers*/
      HI_S32      x;/**<x coordinate absolute*/
      HI_S32      y;/**<y coordinate absolute*/
      HI_U32     pressure;/**<is press on screen: 0, 1*/
      HI_U32     timeStamp;/**<time stamp*/
  } HIGV_TOUCH_INPUTINFO_S;
  HI_GV_Register_TouchDevice
  ```

  This interface is used to register related callback functions and transmit read touchscreen data to the HiGV so that the HiGV can identify related gestures.

[Solution]

From the analysis, you can obtain the following information:

- After the **touchpadadp.c** file is added, you need to check whether the HI_GV_TOUCH_ReadData function is correctly called. You can add related printing information to the function and print the received data through the serial port. If no data is printed after HI_GV_TOUCH_ReadData is called, check whether the enable flag bit interface HI_HAL_TOUCHPAD_Enable is called. Ensure that the s_bEnableFlg flag bit is enabled.
- If there is data output after the HI_GV_TOUCH_ReadData function is called, check whether the output data is correct. If **id** is **0**, it indicates a single-finger touch. If **id** is **1**, it

indicates a 2-finger touch. **x** and **y** represent the horizontal and vertical coordinates of the touch point, respectively. The **pressure** value indicates the pressure status of the touchscreen. When a finger touches the screen, the **pressure** value is always **1**. When the finger leaves the screen, the pressure value is **0**. (If the read pressure value is changing but not **0** when the finger moves on the screen, the pressure value needs to be converted to **1** when the driver reports data.)

1.1.16 What Do I Do If the Text Is Not Properly Displayed?

[Description]

The text content set in the .xml file cannot be properly displayed on the UI.

[Analysis]

The possible causes are as follows:

- The width and height of the text content are greater than or equal to that of the space set for the text on the control. For example, the font size is 30, but the height of the space set for the text on the control is 25. In this case, the text content cannot be displayed.
- The specified font library file directory does not contain the font library file configured in the .xml file.
- The font library file does not support the configured font format. For example, the font library file supports only Chinese and English, while the configured font format in the code is Korean.
- The encoding format of the .xml language file is not UTF-8.

[Solution]

- Ensure that the width and height of the text are smaller than that of the space set for the text on the control.
- Place the corresponding font library file in the specified font library file directory.
- Ensure that the font library file supports the configured font format in the code.
- Set the encoding format of the .xml language file to UTF-8.

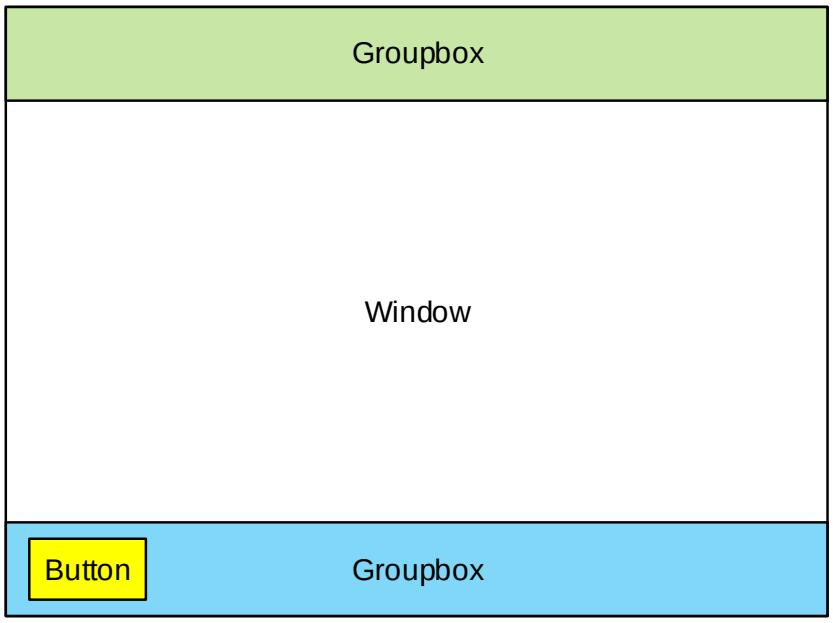
1.2 HiGV Typical Scenarios

1.2.1 Method of Improving Button Control Experience on Small-sized Touchscreens

[Description]

Figure 1-1 shows a typical scenario of a motion DV main UI. The background is a window, and the top and bottom areas are Group Box controls that contain button controls. To simplify the procedure, there is only one button in the figure below. As the screen size is small, the layout of the button control is correspondingly smaller. Therefore the touch experience of the button on screen is poorer and misoperations may occur.

Figure 1-1 Typical main UI



[Analysis]

In addition to touching the button control, add another scenario, namely, allow the touch operation to still respond if you keep the finger on the screen after touching the button and then leave the screen after moving for a while.

[Solution]

Step 1 Register an event.

Register an **ontouchaction** callback function on the xml layout of the window. The **ontouchaction** event is the basis of touch gestures and can receive all gesture messages, including messages of HIGV_TOUCH_START (touch event), HIGV_TOUCH_END (leave event), and HIGV_TOUCH_MOVE (move event).

Step 2 Obtain the handle to the top-layer control.

When a finger touches the screen, the handle to the top-layer control corresponding to the current coordinate can be obtained by calling the HI_GV_Widget_GetWidgetByPos_TouchDevice interface.

Step 3 Add a flag.

Check whether the top-layer control is a button control. If yes, add a flag to the button control.

Step 4 The control responses to the leave action.

The finger keeps moving to move on the screen, but the button control does not respond. When the finger leaves the screen, the HIGV_TOUCH_END event can be parsed. In this case, the asynchronous message interface can be called to send the event notification to the flagged button control. Then the button control responses correspondingly.

For Sample code implementation, see the **middleware/sample/higv/scene_sample** directory.

----End

1.2.2 Method of Improving Fling Gesture Sensitivity on Touchscreens

[Description]

There is a typical scenario on the main UI of the touchscreen, namely fling up, down, left, and right on the main UI to display various interfaces or switch to other windows. General the processing method is registering a callback function for the **ongesturefling** event on the xml control of the window, then determining the action direction inside the callback function, and making response to the fling operation. Here the problem is that the fling event must be triggered at a certain speed, namely, the response appears only when the finger flings quickly on the touchscreen. If the fling speed is insufficient, it will not reach the threshold of triggering the fling event and no response will appear, which may not meet your expectations.

[Analysis]

To improve the fling gesture sensitivity on the main UI, you can perform analysis and combination based on the basic **ontouchaction** touch event, namely, determining whether the fling action is implemented through actions of touch, move, and leave.

[Solution]

Step 1 Register an event.

Register an **ontouchaction** callback function on the xml layout of the window.

Step 2 Obtain the handle to the top-layer control.

When a finger touches the screen, the handle to the top-layer control corresponding to the current coordinate can be obtained by calling the HI_GV_Widget_GetWidgetByPos_TouchDevice interface.

Step 3 Add a flag.

Check whether the top-layer control is a control on the main window. If yes, add a flag to the window control and record the coordinate information of the touch location.

Step 4 Obtain the event information about leaving the screen.

The finger keeps moving on the screen. When the finger leaves the screen, the HIGV_TOUCH_END event can be parsed and the coordinate information of the leave position is recorded.

Step 5 Parse the coordinates.

Implement difference calculation for the coordinate information of the touch and leave positions, get the movement distance in both horizontal and vertical directions, compare the coordinate values, and determine the movement direction. In this way, you can get the fling direction according to the two groups of coordinate information.

For Sample code implementation, see the **middleware/sample/higv/scene_sample** directory.

----End

1.2.3 Control Selection on the Settings UI of Touchscreens

[Description]

Figure 1-2 shows a typical parameter setting UI of motion DV or EDR products. Such UI supports parameter selection by gestures of sliding up and down. The right side of the control is bound to a scroll bar for sliding and displaying slide progress. Currently the HiGV has two types of controls (List Box and Scroll View) that can achieve the similar effect. The following describes differences and selection of the two controls.

Figure 1-2 Typical settings UI

[Analysis]

- For details about the List Box and Scroll View controls, see section 4.7 "List Box" and 4.24 "Scroll View" in *HiGV Development Guide*.
- List Box control, which can be bound to data models and supports batch import of data, are applicable to scenarios with a large amount of data in single data types.
- Scroll View is a container control, in which child controls such as buttons, labels and images can be placed, and can carry controls in redundant types.
- Both List Box and Scroll View controls can be bound to the Scroll Bar control, and their usages are basically the same.

[Solution]

- If the UI for setting the interface design is complex, it is recommended that you use the Scroll View control to achieve diversified UI changes.
- If the settings interface contents are simple and mainly texts with many settings items, it is recommended that you use the List Box control, which requires setting only for once in the **.xml** layout file and supports List Box control sharing by multiple pages. During page switching, call the interface of data refresh to synchronize data only for once, which generates less workload.

1.2.4 Method of Designing the Typical Menu UIs of Motion DV Products

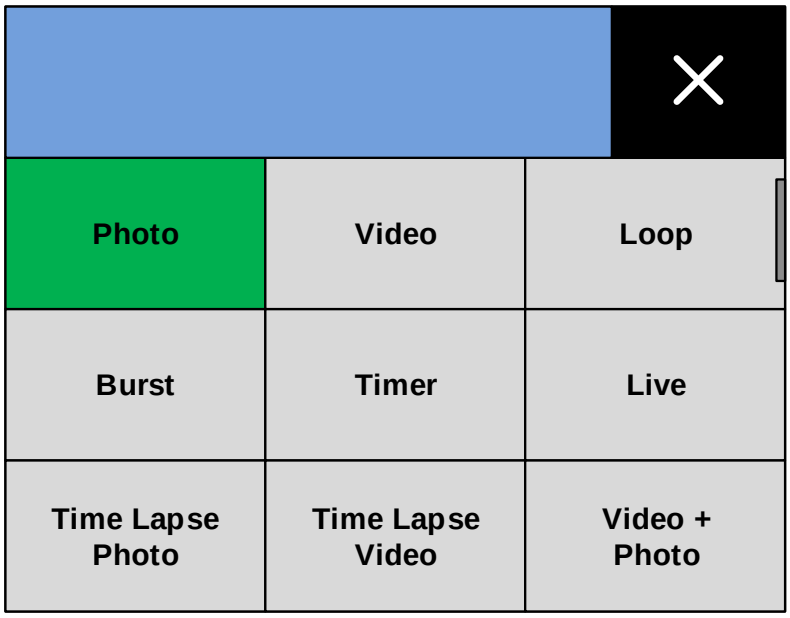
[Description]

This topic describes the UI design methods for typical menus of motion DV products.

[Analysis]

Figure 1-3 shows the typical menu UI of a typical motion DV product.

Figure 1-3 Typical menu UI



[Solution]

- Create a Group Box control and a Scroll View control inside the window, with the former control on the top of the window and the latter control below the former one.
- The Group Box control is a container control, in which buttons and labels can be placed for closing and opening interfaces and displaying information.
- The Scroll View control is a container control, in which buttons can be placed for menu display. Such buttons require settings of **Normalskin** and **Activeskin** to present the selection status of buttons.
- A Scroll View control needs to be bound to one Scroll Bar control to display the slide progress.

1.2.5 Method of Setting Scroll Bar Control Skins

[Description]

This topic describes methods for setting two types of scroll bar skins.

[Analysis]

Just like other controls, the scroll bar skin types are classified into **pic** and **color**, which are configured in the **.xml** skin file. The HiGV control skin consists of nine squares: top, bottom, left, right, left top, right top, bottom left, right bottom, and middle. For setting skins of the color type, generally skin configuration for the middle square is required only. While for setting skins of the pic type, operations must be performed according to the actual situation. Scroll Bar controls are divided into horizontal and vertical ones by the direction, and controls in different directions need skins of different formats.

[Solution]

For horizontal scroll bars, generally skin configurations for the left, middle, and right squares are required only. The skin in the middle square is a rectangle, and skins in the left and right

squares are arrow images pointing to the left and right. Figure 1-4 shows the three skin images in the left, middle, and right.

Figure 1-4 Example of horizontal skins

The following shows an example of configuration in the **.xml** file:

```
<skin
    id="horizontal_skin "
    type="pic"
    isnodrawbg="no"
    linewidth=""
    ltopidx=""
    toplineidx=""
    rtopidx=""
    llineidx="./res/pic/left.png"
    bgidx="./res/pic/middleware.png"
    rlineidx="./res/pic/right.png"
    lbtmidx=""
    rbtmidx=""
    btmlineidx=""
fgidx="0xFFFFFFFF" />
```

- For vertical scroll bars, generally skin configurations for the top, middle, and bottom squares are required only. The skin in the middle square is a rectangle, and skins on the top and bottom squares are arrow images pointing to the top and bottom. Figure 1-5 shows the three skin images on the top, middle, and bottom.

Figure 1-5 Example of vertical skins

The following shows an example of configuration in the **.xml** file:


```
<skin
    id="vertical_skin "
    type="pic"
    isnodrawbg="no"
    linewidth=""
    ltopidx=""
    toplineidx="./res/pic/top.png"
    rtopidx=""
    llineidx=""
    bgidx="./res/pic/middleware.png"
    rlineidx=" "
    lbtmidx=""
    rbtmidx=""
    btmlineidx="./res/pic/button.png"
fgidx="0xFFFFFFFF" />
```

For Sample code implementation, see the **middleware/sample/higv/scene_sample** directory.

1.2.6 Method of Setting Suspending Scroll Bar Controls

[Description]

This topic describes the method of setting suspending status for Scroll Bar controls.

[Analysis]

List Box and Scroll View controls can be bound to scroll bars. In touchscreen scenarios, the experience will be better if the scroll bar suspends over the bottom-layer control.

[Solution]

Take List Box controls as an example. The following describes methods of setting scroll bars to suspending and non-suspending, respectively. The difference between the two methods lies in the settings of width and height in the **.xml** layout file.

- Layout of non-suspending setting

```
    <scrollbar
    id="listbox_scrollbar"
    top="0"
    left="310"
    width="10"
    height="240"
    normalskin="common_skin"
    sliders="scorllbar_res_skin"
    ……
    />

    <listbox
    id="MENUSET_LISTBOX"
    top="0"
```



```
    left="0"
    width="310"
    height="240"
    isrelease="yes"
    scrollbar="listbox_scrollbar"
    normalskin=" common_skin"
    activeskin="common_skin"
    highlightskin=""
    font="simhei_font_text"
……
/>
```

- Layout of suspending setting

```
    <scrollbar
    id="listbox_scrollbar"
    top="0"
    left="310"
    width="10"
    height="240"
    normalskin="common_skin"
    slidres="scorllbar_res_skin"
    transparent="yes"
    ……
    />

    <listbox
    id="MENUSET_LISTBOX"
    top="0"
    left="0"
    width="320"
    height="240"
    isrelease="yes"
    scrollbar="listbox_scrollbar"
    normalskin=" common_skin"
    activeskin="common_skin"
    highlightskin=""
    font="simhei_font_text"
……
/>
```

- Comparing the layouts for non-suspending and suspending scroll bars, we can conclude that the major difference is the width configuration of List Box controls, namely, non-suspending scenarios require space reservation for the scroll bar, while suspending scenarios do not require any space reservation.

- Scroll bar suspending requires the background attribute to be transparent, namely transparent = "yes", whereas non-suspending layout generally does not require the attribute of transparent background.
- The situation of Scroll View controls is basically the same as that of List Box controls. Note that Scroll View controls are horizontal or vertical, while List Box controls are vertical only. Correspondingly, for setting horizontal Scroll View controls, only the height in the layout needs to be modified.

For Sample code implementation, see the **middleware/sample/higv/scene_sample** directory.

1.2.7 Time and Date Function Development for Touchscreens

[Description]

The time and date UI function for touchscreens requires content switching through the sliding gesture.

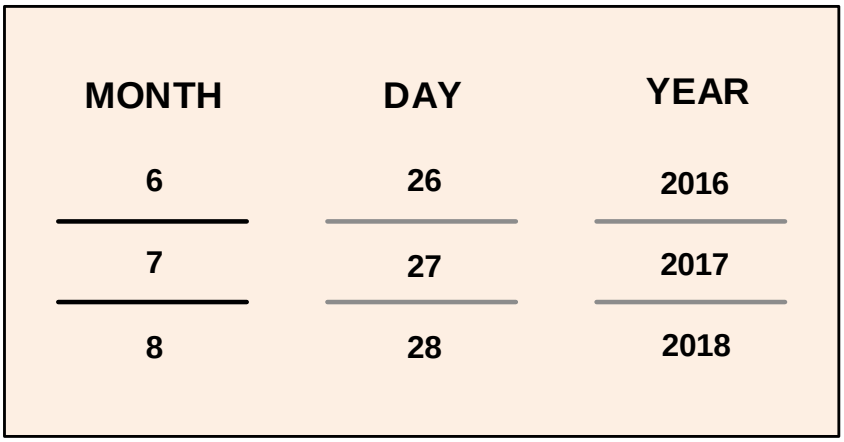
[Analysis]

It is recommended that a Wheel View control combination is used for achieving the similar effect. You can combine three Wheel View controls side-by-side to achieve the effect of mm/dd/yyyy or hh:mm a.m. or p.m.

[Solution]

Figure 1-6 shows a typical date control on the touchscreen. Three labels or images work as titles in upper part, and three Wheel View controls are laid side-by-side below the titles. Different contents are set to the Wheel View controls.

Figure 1-6 Typical date control



For Sample code implementation, see the **middleware/sample/higv/widget_sample** directory.

1.2.8 Method of Page Flipping on Scroll Grid Controls in Non-touchscreen Scenarios

[Description]

This topic describes interfaces for page flipping on view list interfaces (Scroll Grid controls) by pressing keys in non-touchscreen scenarios.

[Analysis]

None

[Solution]

- Go to the next page (page down or right): HI_GV_ScrollGrid_PageForward
- Go to the previous page (page up or left): HI_GV_ScrollGrid_PageBackward
- Directly go to the first page: HI_GV_ScrollGrid_MoveToOrigin
- Directly go to the last page: HI_GV_ScrollGrid_MoveToLast
- Directly go to the page of a specified cell: HI_GV_ScrollGrid_SetSelCell

For details about the interface description, see the header file **hi_gv_scrollgrid.h**.

1.2.9 Method of Setting Multiple Languages

[Description]

This topic describes methods of setting multiple languages on the GUI.

[Analysis]

The HiGV supports multiple languages. You can switch languages of different countries by calling an interface. In practical applications, the main methods of language setting are as follows:

- Different languages share a font library.
- Multiple font libraries are used for different languages. Possibly one or more languages uses exclusive font libraries.

[Solution]

- For font library sharing by different languages, see the setting method in section 3.5.3 "Configuring Resources in the XML File" of *HiGV Development Guide*. The setting method is simple.
- For usage of exclusive font libraries for different languages, font set interfaces need to be called for adding fonts. The HI_GV_FontSet_Create interface is used for creating a font set, and the HI_GV_FontSet_AddFont interface is used for adding font handles to the font set. The following shows a code example. First of all, create a handle to font set, hFontSet. After the handle creation is complete, add fonts FONT20 and FONT_KAI_20 to the font set. The two fonts can come from two font libraries. The FONT20 contains Chinese and Japanese, and the FONT_KAI_20 contains English. In practical applications, HI_GV_Lan_Change can be called for switching fonts in different formats.

```
HI_HANDLE hFontSet;
    if(HI_SUCCESS == HI_GV_FontSet_Create(&hFontSet))
    {
        (HI_VOID)HI_GV_FontSet_AddFont(FONT20, "zh;ja", hFontSet);
         (HI_VOID)HI_GV_FontSet_AddFont(FONT_KAI_20, "en", hFontSet);
    }
```

For Sample code implementation, see the **middleware/sample/higv/scene_sample** directory.

1.2.10 Parameter Parsing of Registering Callback Functions for Touchscreen Messages

[Description]

This topic describes how to parse parameters of callback functions that correspond to touchscreen message events.

[Analysis]

There are two types of touch messages: basic touch event messages and the touch gesture event messages. These messages IDs corresponding to the two types of messages are as follows:

- Basic touch event messages: HIGV_MSG_TOUCH
- Gesture event messages: HIGV_MSG_GESTURE_TAP (tap and leave gesture), HIGV_MSG_GESTURE_LONGTAP (touch and hold gesture), light HIGV_MSG_GESTURE_FLING (fling gesture), and HIGV_MSG_GESTURE_SCROLL (slide gesture)

The two types of messages correspond to different data structures and parsing methods.

[Solution]

The following shows an example of parameter parsing for the callback function of basic touch event messages.

```
HI_S32 CSlideUnlock::OnTouchAction(HI_U32 wParam, HI_U32 lParam)
{
    HI_S32 TouchLeft = 0;
    HI_S32 TouchTop = 0;
    HIGV_TOUCH_EVENT_S touchEvent = {0};
    HIGV_TOUCH_E type = HIGV_TOUCH_BUTT;

    HIGV_MemSet(&touchEvent, 0x0, sizeof(touchEvent));
    HIGV_MemCopy(&touchEvent, (HIGV_TOUCH_EVENT_S*)lParam,
sizeof(touchEvent));

    TouchLeft = touchEvent.last.x;
    TouchTop = touchEvent.last.y;
    type = touchEvent.last.type;

    switch (type)
    {
        case HIGV_TOUCH_START:
        {
            (HI_VOID)TouchStart(TouchLeft, TouchTop);
            break;
        }
        case HIGV_TOUCH_END:
        {
```



```
            (HI_VOID)TouchEnd(TouchLeft, TouchTop);
            break;
        }
        case HIGV_TOUCH_MOVE:
        {
            (HI_VOID)TouchMove(TouchLeft, TouchTop);
            break;
        }
        default:
            break;
    }
    return HIGV_PROC_GOON;
}
```

Description:

1. The corresponding message ID is HIGV_MSG_TOUCH.
2. The data structure of the touch event is HIGV_TOUCH_EVENT_S. For details about the data structure, see section 2.1.10 "Gesture" in *HiGV API Reference*.
3. The **lParam** parameter is the pointer to the touch event data structure and points to a segment of memory data. The **wParam** parameter is the length of the touch event data structure, namely, the length of the content to which **lParam** points.
4. HIGV_TOUCH_START (touch event), HIGV_TOUCH_END (leave event), and HIGV_TOUCH_MOVE (move event) are three most basic message events included in touch messages. All gesture events are combinations and encapsulations based on the three events.
5. The four gesture message types correspond to data structures of HIGV_GESTURE_TAP_S, HIGV_GESTURE_LONGTAP_S, HIGV_GESTURE_FLING_S, and HIGV_GESTURE_SCROLL_S. For details about data structure descriptions, see section 2.1.10 "Gesture" in *HiGV API Reference*. The parameter parsing methods are basically the same as that of the HIGV_MSG_TOUCH, so you only need to replace corresponding data structures.

