|+ 4˜

|+ 4˜

ü rL h*ü È*ü T T á ¾ à ï à Ê ;> Ô þ |+ È5à LÔ?U Ú î þ |+ G6(CK 9 J> ê > ;> È ¹9‹ k È á ,X |+ p Ä ¹ mGEff é 9 Z |+ 4˜ È*ü Ã ¹EîE› ï Î J> (Parallel) ê > (Sequential) 4˜ 9E' ;> Ô4˜ |+ ,XLÔ" Ä

- J> 4˜ |+
  J> 4˜,X(M&• ' |+ 4˜ ;> Ê4˜ Y Ý |+ FÑ à Ê>•A× z È><)„,X p E- o |+ à Ê üE¯ > È4˜ |+ ,X E¤> ÊKÈ+ ÔKS,XFw þ |+ E¤> ÊKÈ ‡ n
- > 4˜ |+
  > 4˜,X(M&• |+ 4˜ ;> Ê4˜ Y |+ Ú>•NN cA× z È><)„,X p E- o |+ Ú â/½ cE¯ > È4˜ |+ ,X E¤> ÊKÈ Ý |+ E¤> ÊKÈ,X ` Ä º ê Ã ¹ Ú Ô þ |+ 4˜ ' 0 Ô þ |+ #Ï t º Ô þ4˜ È JA× z?˜ í à Þ Ä

Ñ D s _

mGEff ¤ o Z V ß ´ þ Ñ D*ü b Õ |+ 4˜,X ï Î ` S*ü Ä J Ñ D s _ V ß Ö

ï Î |+ 4˜ Ñ D

```
MGEFF_ANIMATION mGEffAnimationCreateGroup(      enum  EffAnimationType type);
```

- – DAÈ â Ö
  - type - |+ 4˜2O _ È Ã ¹ MGEFF_PARALLEL Ä J> Å ` MGEFF_SEQUENTIAL Ä > Å ` Ô ×
- E¨ ² AÈ â Ö
  - handle - ä sE¨ ² |+ 4˜ ¹ ˜ È ú íE¨ ² NULL ×

#Ï t |+ 4˜ Ñ D

```
/**
*   Ú Ô þ |+  t 9 |+ 4˜
```

```
*/
void   mGEffAnimationAddToGroup(MGEFF_ANIMATION group, MGEFF_ANIMATION a
```

z – DAÈ â Ö
  { group - |+ 4˜ ¹ ˜ ×
  { animation - Ú t 9 group ,X |+ ¹ ˜ ×
z E¨ ² AÈ â Ö
  { void - ×

Ô&• V É

¢ Ñ D s _ Ã ¹,ß È |+ 4˜,X ¹ ˜ ` |+ ¹ ˜ à ,X2O _ È !M6 Ÿ4¡E› mGEff ,XA'Au ÞG› *ü Z Ô oM6 å ÍB5 ñ Ç È |+ 4˜ ÍB5 ¢ |+ ÍB5 "4» S"5à 9 Ä |+ 4˜ 3 Ô/¡ |+ È ! Ÿ4¡E›,X | + 2 û Ã Þ ß [ `E¤> • ã,Ì G,X Ñ D Ã 0*ü b |+ 4˜ ÈE- Ô&• Ú ü ß Ô8V,X/ _)/ Î 9 Ä

S*ü/ _

J> |+ 4˜

```
#include    <stdio.h>
#include    <string.h>

#include    <minigui/common.h>
#include    <minigui/gdi.h>
#include    <minigui/window.h>
#include    <minigui/minigui.h>

#include    <mgeff/mgeff.h>

/*********************************************************************
#define    CAPTION    "    group_animation_parallel        "
#define    BAR_HEIGHT 50
#define    DURATION   (5 * 1000)
#define    START_VAL  0x00
#define    END_VAL    0xFF
#define    ANIMATION_NUM 3

/*********************************************************************
static     int   g_color[ANIMATION_NUM] = { 0 };

/*********************************************************************
/* main window proc */
static     int   mainWindowProc (HWND hWnd,     int    message, WPARAM wParam, LPAR
/* draw a frame */
static     void   draw_frame (HWND hWnd);
/* callback function called when property change */
static     void   property_callback (MGEFF_ANIMATION handle, HWND hWnd,          int
```

```
/* create and run an animation */
static      int    do_animation (HWND hWnd);

/************************************************************************
int    MiniGUIMain (      int    argc,      const     char   *argv[])
{
        HWND hMainHwnd;
        MAINWINCREATE createInfo;
        MSG msg;

                #ifdef     _MGRM_PROCESSES
        JoinLayer (NAME_DEF_LAYER, "              animation    ", 0, 0);
                #endif

        createInfo.dwStyle = WS_VISIBLE | WS_BORDER | WS_CAPTION;
        createInfo.dwExStyle = WS_EX_NONE;
        createInfo.spCaption = CAPTION;
        createInfo.hMenu = 0;
        createInfo.hCursor = GetSystemCursor (0);
        createInfo.hIcon = 0;
        createInfo.MainWindowProc = mainWindowProc;
        createInfo.lx = 0;
        createInfo.ty = 0;
        createInfo.rx = 240;
        createInfo.by = 320;
        createInfo.iBkColor = COLOR_lightwhite;
        createInfo.dwAddData = 0;
        createInfo.hHosting = HWND_DESKTOP;

        hMainHwnd = CreateMainWindow (&createInfo);

                if    (hMainHwnd == HWND_INVALID) {
                        return     -1;
        }

        ShowWindow (hMainHwnd, SW_SHOWNORMAL);

                while    (GetMessage (&msg, hMainHwnd)) {
                TranslateMessage (&msg);
                DispatchMessage (&msg);
        }

        MainWindowThreadCleanup (hMainHwnd);

                return     0;
}

/************************************************************************
```

```
static int mainWindowProc (HWND hWnd, int message, WPARAM wParam, LPAR
{
    switch (message) {
        case MSG_CREATE:
        /* init mgeff library */
    mGEffInit ();

    do_animation (hWnd);
        break ;

        case MSG_PAINT:
    draw_frame (hWnd);
        break ;

        case MSG_CLOSE:
    DestroyMainWindow (hWnd);
    PostQuitMessage (hWnd);

        /* deinit mgeff library */
    mGEffDeinit ();
        break ;

        default :
        break ;
    }

    return DefaultMainWinProc (hWnd, message, wParam, lParam);
}

static void draw_frame (HWND hWnd)
{
    HDC dc;
    RECT rc;

    int client_w, client_h;
    char str[ANIMATION_NUM][64];

    dc = BeginPaint (hWnd);

    /* get client rect */
    GetClientRect (hWnd, &rc);

    client_w = RECTW (rc);
    client_h = RECTH (rc);          // - BAR_HEIGHT) * g_value / (END_VAL -

    /* draw */
    /* red */
    SetBrushColor (dc, RGB2Pixel (dc, g_color[0], 0, 0));
```

```
        FillBox (dc, 0, client_h * 0 / ANIMATION_NUM, client_w, client

        sprintf (str[0], "                %d", g_color[0]);
        TextOut (dc, 0, client_h * 0 / ANIMATION_NUM, str[0]);

            /* green */
        SetBrushColor (dc, RGB2Pixel (dc, 0, g_color[1], 0));
        FillBox (dc, 0, client_h * 1 / ANIMATION_NUM, client_w, client

        sprintf (str[1], "                %d", g_color[1]);
        TextOut (dc, 0, client_h * 1 / ANIMATION_NUM, str[1]);

            /* blue */
        SetBrushColor (dc, RGB2Pixel (dc, 0, 0, g_color[2]));
        FillBox (dc, 0, client_h * 2 / ANIMATION_NUM, client_w, client

        sprintf (str[2], "                %d", g_color[2]);
        TextOut (dc, 0, client_h * 2 / ANIMATION_NUM, str[2]);

        EndPaint (hWnd, dc);
}

static    void   property_callback (MGEFF_ANIMATION handle, HWND hWnd,          int
{
        g_color[id] = *value;

        InvalidateRect (hWnd, NULL, TRUE);
}

static    int    do_animation (HWND hWnd)
{
        MGEFF_ANIMATION animation[ANIMATION_NUM];
        MGEFF_ANIMATION group_animation;

            int    duration;;
            int    start_val;
            int    end_val;

            int    i;

            /* set value */
        duration = DURATION;
        start_val = START_VAL;
        end_val = END_VAL;

            /* create an animation group */
        group_animation = mGEffAnimationCreateGroup (MGEFF_PARALLEL);
            //group_animation = mGEffAnimationCreateGroup (MGEFF_SEQUENTIA
```

```
        /* create and animation and add it to a group */
        for   (i = 0; i < ANIMATION_NUM; i++) {
                /* create animation */
            animation[i] = mGEffAnimationCreate ((                    void   *) hWnd, (      v
                /* set property */
                /* duration */
            mGEffAnimationSetDuration (animation[i], duration);

                /* start value */
            mGEffAnimationSetStartValue (animation[i], &start_val)

                /* end value */
            mGEffAnimationSetEndValue (animation[i], &end_val);

                /* add animation to group */
            mGEffAnimationAddToGroup (group_animation, animation[i
    }

        /* running */
    mGEffAnimationAsyncRun (group_animation);

        /* wait animation end */
    mGEffAnimationWait ((              void   *) &hWnd, group_animation);

        /* delete the animation object */
    mGEffAnimationDelete (group_animation);

        return    0;
}
```

> |+ 4˜

¾?U Ú Þ þ/ _ ,X ï Î |+ 4˜ Ñ DA×*ü Ó 6 ä ßM6E- ¹ È Ã ¹ r)„ Ô þ > |+ 4˜ Ö

```
/* ... */
group = mGEffAnimationCreateGroup(MGEFF_SEQUENTIAL);
/* ... */
```

0´ ã4§

-- XuBinWang - 12 Jan 2011