Background Images

The Device Kit − Table of Contents

Background Images

Declared in: be/be_apps/Tracker/Background.h

Tracker lets you put background images on each workspace on your desktop and in the background of individual Tracker folder windows.

Currently, Tracker only paints the background of a folder window when the window is in icon (or mini−icon) view mode.

Background Attributes

Information about the background image for a folder is stored as a flattened BMessage in the folder's **B_BACKGROUND_INFO** file attribute. The **B_DESKTOP_FOLDER** directory holds the background info for the workspaces (one flattened message per workspace).

The **B_BACKGROUND_INFO** attribute's data is a flattened BMessage with these fields:

B_BACKGROUND_IMAGE	**B_STRING_TYPE**	The full path to the image. The image needs to be in a format understood by one of the translators (see the Translation Kit) installed on the system.
B_BACKGROUND_MODE	**B_INT32_TYPE**	Specifies how the background is placed (tiled, centered, etc). See the list of mode constants after this table.
B_BACKGROUND_ORIGIN	**B_POINT_TYPE**	If **B_BACKGROUND_MODE** is **B_BACKGROUND_MODE_USE_ORIGIN**, the image will be drawn with the upper left corner at this point (which can be anything, even negative).
B_BACKGROUND_ERASE_TEXT	**B_BOOL_TYPE**	If **true** the icon label will have its background erased to the window's background color, otherwise the label will be drawn on top of the image (which could make it difficult to read).
B_BACKGROUND_WORKSPACES	**B_UINT32_TYPE**	A bitmask indicating which workspaces the image should be drawn on. This only applies to Desktop images.

The **B_BACKGROUND_MODE** values are:

- **B_BACKGROUND_MODE_USE_ORIGIN**. Draw the image once at the location specified by **B_BACKGROUND_ORIGIN**, relative to the upper−left corner of the window or Desktop.
- **B_BACKGROUND_MODE_CENTERED** (only works on Desktop) Center the image.
- **B_BACKGROUND_MODE_SCALED** (only works on Desktop) Scale the image to fit.
- **B_BACKGROUND_MODE_TILED** Tile the image to cover the entire window/Desktop.

You can write multiple sets of these message fields into the Desktop's **B_BACKGROUND_INFO** attribute message in order to put different images on different workspaces. The fields are correlated by index. Don't put different **B_BACKGROUND_IMAGE** paths onto the same workspace though, the behavior is undefined.

For example, this is OK:

```
...
/* ALMOST_ALL_WORKSPACES is every workspace except the first */
const int32 ALMOST_ALL_WORKSPACES = 0xffffffff - 0x00000001;
const int32 OTHER_WORKSPACE = 0x00000001;
...
BMessage backgrounds;

/* Add the first image to ALMOST_ALL_WORKSPACES */
backgrounds.AddString( B_BACKGROUND_IMAGE, "/boot/home/nice.jpg" );
backgrounds.AddInt32( B_BACKGROUND_WORKSPACES, ALMOST_ALL_WORKSPACES );
backgrounds.AddInt32( B_BACKGROUND_MODE, B_BACKGROUND_MODE_TILED );
backgrounds.AddPoint( B_BACKGROUND_ORIGIN, BPoint( 0.0, 0.0 ) );
backgrounds.AddBool( B_BACKGROUND_ERASE_TEXT, true );

/* Add the second image to the OTHER_WORKSPACE */
backgrounds.AddString( B_BACKGROUND_IMAGE, "/boot/home/nicer.jpg" );
backgrounds.AddInt32( B_BACKGROUND_WORKSPACES, OTHER_WORKSPACE );
backgrounds.AddInt32( B_BACKGROUND_MODE, B_BACKGROUND_MODE_SCALED );
backgrounds.AddPoint( B_BACKGROUND_ORIGIN, BPoint( 0.0, 0.0 ) );
backgrounds.AddBool( B_BACKGROUND_ERASE_TEXT, true );
...
```

This is *not* OK, and results in undefined behaviour:

```
...
const int32 ALL_WORKSPACES = 0xffffffff;
...
BMessage backgrounds;

backgrounds.AddString( B_BACKGROUND_IMAGE, "/boot/home/nice.jpg" );
backgrounds.AddInt32( B_BACKGROUND_WORKSPACES, ALL_WORKSPACES );
```

```
/* ... other B_BACKGROUND_* data here ... */

backgrounds.AddString( B_BACKGROUND_IMAGE, "/boot/home/nicer.jpg" );
backgrounds.AddInt32( B_BACKGROUND_WORKSPACES, ALL_WORKSPACES );
/* ... other B_BACKGROUND_* data here ... */
...
```

The **B_BACKGROUND_INFO** attribute's contents might be extended in the future (but will remain compatible with the existing attribute data). Be prepared to preserve any data you don't recognize by reading in the existing **B_BACKGROUND_INFO** attribute (if one exists).

Special Folders

There are two special folders you can apply attributes to:

B_DESKTOP_DIRECTORY	Applying the **B_BACKGROUND_INFO** attribute to **B_DESKTOP_DIRECTORY** (see **find_directory()** in the Storage Kit) will change the Desktop's background image (but see below to find out how to tell the Tracker to update the image).
Tracker/DefaultFolderTemplate inside **B_USER_SETTINGS_DIRECTORY**	Applying the **B_BACKGROUND_INFO** attribute to this directory sets the default folder background. Any folder that doesn't have its own **B_BACKGROUND_INFO** will display this image.

Notifying Tracker

Once you've written an appropriate **B_BACKGROUND_INFO** attribute, you'll want to tell the Tracker to pick up the new background image.

Send the Tracker's main looper (remember, the Tracker's signature is "application/x−vnd.Be−TRAK") a BMessage with a *what* of **B_RESTORE_BACKGROUND_IMAGE** to tell it to reload the desktop background image.

Sending the same message to the looper for a specific Tracker window will tell *that window* to reload its background image. This is a little trickier because you need to find the right window.

Refreshing All Windows

Taking the shotgun approach, you can tell all of the open Tracker windows to repaint their backgrounds by using Tracker scripting (see Scripting in the Application Kit). You'll need to do this if you change the default folder background image.

This code asks the Tracker for windows that have a "Poses" property and sends them a **B_RESTORE_BACKGROUND_IMAGE**. This updates every Tracker window that's displaying a folder's contents.

```
BMessenger tracker( "application/x−vnd.Be−TRAK" );

int32 i = 0;
BMessage reply;
int32 err;

do {
    /* scripting message */
    BMessage msg( B_GET_PROPERTY );

    /* look at the "Poses" in every Tracker window */
    msg.AddSpecifier( "Poses" );
    msg.AddSpecifier( "Window", i++ );

    reply.MakeEmpty();

    tracker.SendMessage( &msg, &reply );

    /* break out of the loop when we're at the end of
     * the windows
     */
    if( reply.what == B_MESSAGE_NOT_UNDERSTOOD &&
        reply.FindInt32( "error", &err ) == B_OK &&
        err == B_BAD_INDEX )
        break;

    /* don't stop for windows that don't understand
     * a request for "Poses"; they're not displaying
     * folders
     */
    if( reply.what == B_MESSAGE_NOT_UNDERSTOOD &&
        reply.FindInt32( "error", &err ) == B_OK &&
        err != B_BAD_SCRIPT_SYNTAX )
        continue;

    BMessenger m;
    if( reply.FindMessenger( "result", &m ) == B_OK) {
        /* m is the messenger of a Tracker window with
         * a folder inside
         */

        m.SendMessage( B_RESTORE_BACKGROUND_IMAGE );
    }

} while( true );
```

Refreshing One Window

If want to refresh the background of the folder you've changed, you scan through the open folders using code similar to the above, until you find the folder with the "Path" you're interested in. Once you've got the right window, you send it a **B_RESTORE_BACKGROUND_IMAGE** message. Using this technique is nicer (from the user's point of view) because only one Tracker window gets updated.

Here's the same code, modified to target a specific path. Assume you've still got the path in question in *the_path* (you just used it as a **B_BACKGROUND_IMAGE**):

```
BMessenger tracker( "application/x-vnd.Be-TRAK" );

/* CHANGE: we'll increment "i" at the start of the loop
 *         now; this lets us use it later when we find
 *         a Window with a Poses
 */
int32 i = -1;
BMessage reply;
int32 err;

do {
    /* CHANGE: */
    i++;

    /* scripting message */
    BMessage msg( B_GET_PROPERTY );

    /* look at the "Poses" in every Tracker window */
    msg.AddSpecifier( "Poses" );
    msg.AddSpecifier( "Window", i ); /* CHANGE */

    reply.MakeEmpty();

    tracker.SendMessage( &msg, &reply );

    /* break out of the loop when we're at the end of
     * the windows
     */
    if( reply.what == B_MESSAGE_NOT_UNDERSTOOD &&
        reply.FindInt32( "error", &err ) == B_OK &&
        err == B_BAD_INDEX )
        break;

    /* don't stop for windows that don't understand
     * a request for "Poses"; they're not displaying
     * folders
     */
    if( reply.what == B_MESSAGE_NOT_UNDERSTOOD &&
        reply.FindInt32( "error", &err ) == B_OK &&
        err != B_BAD_SCRIPT_SYNTAX )
        continue;

    /* CHANGE: everything from here down is new */

    /* make a messenger to this window */
    BMessenger m;
    if( reply.FindMessenger( "result", &m ) != B_OK) continue;

    /* found a Window with a Poses, ask for its Path */
    msg.MakeEmpty();
    msg.what = B_GET_PROPERTY;
    msg.AddSpecifier( "Path" );
    msg.AddSpecifier( "Poses" );
    msg.AddSpecifier( "Window", i );

    reply.MakeEmpty();

    tracker.SendMessage( &msg, &reply );

    /* go on with the next if this din't have a Path */
    if( reply.what == B_MESSAGE_NOT_UNDERSTOOD ) continue;

    /* dig out the Path */
    entry_ref ref;
    if( reply.FindRef( "result", &ref ) == B_OK ) {
        BEntry ent( &ref );
        BPath path( &ent );

        /* these are not the Paths you're looking for */
        if( strcmp( the_path, path.Path() ) ) continue;
    }

    /* if you got this far, you have a messenger to a valid Tracker
     * window displaying a folder with the path you were after,
     * so tell it to refresh its background
     */
    m.SendMessage( B_RESTORE_BACKGROUND_IMAGE );

} while( true );
```