



Sky Operating System

SkyOS 5.0 GUI (WindUI)
Principals and Functionality
Copyright © 1996 - 2003 by Robert Szeleney



Table of contents

TABLE OF CONTENTS.....	2
INTRODUCTION AND OVERVIEW.....	3
WHAT IS THE NEW GUI ABLE TO DO:	3
GUI TECHNIQUES	4
CLIPPING	4
OFFSCREEN WINDOW BITMAPS	4
WINDOW FORM BITMAP	4
PERFORMANCE OPTIMIZING	5

Introduction and overview

To meet the requirements of the new WindUI GUI and to make it possible to use modern graphic effects a total GUI redesign was necessary.

As you may know, the SkyOS GUI consist of two parts. The kernel part (GiWM) which handles all low level window management and graphic drawing, and the user part (SkyGI) which includes widgets and message handling.

The redesign was done for the kernel part mostly.

What is the new GUI able to do:

The WindUI features a lot of more graphic effects than the old one:

- Alpha transparency
- Translucent alpha blended windows
- Freeform windows
- Rounded windows
- Window animation
- Window shadows
- Optionally OpenGL rendered windows
- Full themeing and skinning support
-

GUI techniques

This chapter will explain how the different used GUI techniques works and what they are used for.

Clipping

Clipping is a very important part of every GUI. With clipping the GUI decides which parts of windows are visible or not.

The clipping manager computes each visible rectangles for every window whenever a window is moved or resized.

So each window has a list of visible rectangles after the clipping process finished. Each drawing operation (line, rectangle, blit,...) checks if the resulting drawing is in or outside the visible clipping regions or whether it must “cut” to this clipping rectangles.

For SkyOS 4.9b the “ClipManager” was totally rebuilt. So the ClipManager supports different visible rectangles like full visible, alpha blended visible, shadow visible, ... now. Because of the knowledge of this different visible types the ClipManager is able choose the best method when redrawing such regions. (This gains a dramatically perform increase).

Offscreen window bitmaps

Since SkyOS 4.9b there is support for Offscreen window bitmaps. When enabling the “Advanced GUI techniques” button in the SystemManager, the WindUI GUI uses different techniques for drawing windows.

The main difference is that all drawing operations aren’t performed into the screen anymore. Instead, all drawing is done into offscreen window bitmaps. This bitmaps may resist in graphics memory or DRAM.

Whenever a window finishes its drawing operations the modified window areas are blitted onto the screen.

Using this Offscreen window bitmaps we get following advantages:

- No application redraw needed whenever a window area is exposed
- Support for alphablended translucent windows
- Support for OpenGL “animated” windows. E.g. when blitting the window bitmap to the screen, the OpenGL display driver can make transformations, rotations are any other OpenGL specific operation on the window. (E.g. showing the windows on a big cube in the middle of the screen)

Window form bitmap

Each window can have an optional window form bitmap. This 1Byte bitmap defines the alpha value for each pixel and also the complete freeform shape.

This window form bitmap is used to generate window shadows and rounded windows. Additionally the user may fully modify it to generate any desired window shape.

Performance Optimizing

As you may know all features like alpha blending, rounded windows, Costs a lot of processor time and may reduce the overall performance.

To meet the individual wishes of users, most GUI effects can be enabled or disabled.

The following list shows which option disables/enables which feature: (These options can be set with the SystemManager and the Theme plugin)

- Alpha blended windows

When enabled:

1. The "paint message" mechanism for windows is disabled. Windows no longer receive any paint message for "dirty" window regions.
2. For each top-level window (if not other requested by an application) a window bitmap is generated and a redraw message is sent. So each window draws itself into an offscreen image.
3. A EndPaint message handler in the kernel is activated which blits the offscreen image to the screen whenever a window has finished updating it's window content.
4. The dirty region manager is activated which keeps track of dirty window regions. This window regions are blitted from the offscreen image to the screen everytime a window finished drawing.
(Optionally specified alpha transparency is used when computing the resulting screen image)

Note: After enabling this option, all application (if not other requested) are drawing into offscreen bitmaps. (Either allocated in VideoRAM or DRAM).

When disabled:

1. The "paint message" mechanism is activated
2. The dirty region manager is deactivated
3. Automatically created offscreen images are destroyed
4. The EndPaint message handler gets deactivated

Note: After disabling this option, all application (if not other requested) are drawing directly onto the screen.

- Window shadows and rounded corners

When enabled and current theme supports these effects:

1. For each frame window a freeform window bitmap is created.
2. Each frame window computes the theme dependent window form and shadow width.
3. The clip manager is requested to keep track of transparent window regions. After this, the clip manager recomputes all visible/invisible/partial visible regions of all windows.

Note: When enabling this option, drawing operations which occur under transparent or shadowed window regions may be slower.

When disable to current theme doesn't support these effects:

Turn off all options listed above.

- Window animation

When enabled:

Installs a kernel "post window create" handler which handles the entire window animation.

Depending on the application/global selected window animation, this handlers draws the window.

E.g. fade in/out, roll in/out,...

As you can see, SkyOS tries to gain a maximum on performance when disabling features. Not only by don't perform unneeded computations, but by replacing the entire code. (E.g. Switching between different clipping managers, redraw managers, device drivers,...)