

Libraries

| |
|----------------------|
| COLLABORATORS |
|----------------------|

| | | | |
|---------------|-----------------------------|----------------|------------------|
| | <i>TITLE :</i> Libraries | | |
| <i>ACTION</i> | <i>NAME</i> | <i>DATE</i> | <i>SIGNATURE</i> |
| WRITTEN BY | | March 28, 2025 | |

| |
|-------------------------|
| REVISION HISTORY |
|-------------------------|

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
| | | | |

Contents

| | | |
|----------|---|----------|
| 1 | Libraries | 1 |
| 1.1 | RKM Libraries: 2 Intuition and the Amiga Graphical User Interface | 1 |
| 1.2 | 2 Intuition and the Graphical User Interface / About User Interfaces | 1 |
| 1.3 | 2 / User Interfaces / Elements of the Graphical User Interface System | 2 |
| 1.4 | 2 / About User Interfaces / Goals of Intuition | 3 |
| 1.5 | 2 Intuition and the Graphical Interface / How the User Sees Intuition | 3 |
| 1.6 | 2 / How the User Sees Intuition / Workbench and Preferences | 4 |
| 1.7 | 2 / How the User Sees Intuition / Intuition's 3D Look | 4 |
| 1.8 | 2 Intuition & Graphical Interface / How an Application Sees Intuition | 5 |
| 1.9 | 2 / How an Application Sees Intuition / Components of Intuition | 5 |
| 1.10 | 2 / How an Application Sees Intuition / Screens and Windows | 5 |
| 1.11 | 2 / How an Application Sees Intuition / Gadgets, Menus And Requesters | 6 |
| 1.12 | 2 // Gadgets, Menus And Requesters / Gadgets | 6 |
| 1.13 | 2 // Gadgets, Menus And Requesters / Menus | 6 |
| 1.14 | 2 // Gadgets, Menus And Requesters / Requesters | 7 |
| 1.15 | 2 // Gadgets, Menus And Requesters / The Intuition Input Event Loop | 7 |
| 1.16 | 2 Intuition and the Graphical Interface / A Simple Intuition Program | 8 |
| 1.17 | 2 / A Simple Intuition Program / Example Intuition Event Loop | 9 |

Chapter 1

Libraries

1.1 RKM Libraries: 2 Intuition and the Amiga Graphical User Interface

Intuition is the collective name for the function libraries, data structures and other elements needed to create a graphical interface for Amiga applications. Programmers use Intuition to perform user interface chores such as opening windows, managing menus, monitoring gadgets, reading the mouse position and so forth.

Newcomers to the Amiga sometimes think of Intuition as the Amiga's operating system but it is not. Intuition is just one component that together with Exec, AmigaDOS, and other subsystems make up the whole operating system. Intuition is the most visible part of the operating system though since it provides the graphical user interface familiar to all Amiga users.

| | |
|-----------------------------|-----------------------------------|
| About User Interfaces | How an Application Sees Intuition |
| How the User Sees Intuition | A Simple Intuition Program |

1.2 2 Intuition and the Graphical User Interface / About User Interfaces

What is a user interface? This sweeping phrase covers all aspects of communication between the user and the computer. It includes the innermost mechanisms of the computer and rises to the height of defining a philosophy to guide the interaction between human and machine. Intuition is, above all else, a philosophy turned into software.

Intuition's user interface philosophy is simple to describe: the interaction between the user and the computer should be consistent, simple and enjoyable; in a word, intuitive. Intuition supplies the tools needed to turn this philosophy into practice.

Implicit in this philosophy is the idea that the user interface should be graphical. A graphical user interface, or GUI, is a visually oriented method of communicating with a computer in which system resources are represented by pictorial symbols that can be manipulated with a pointing device such as a mouse. Other types of user interfaces are possible such as the Amiga's Shell in which text commands are entered by typing them at

the keyboard. For more information about user interfaces, refer to the Amiga User Interface Style Guide.

Elements of the Amiga Graphical User Interface System
Goals of Intuition

1.3 2 / User Interfaces / Elements of the Graphical User Interface System

There is more to the Amiga user interface than Intuition. To build a complete user interface, application writers need to understand these other elements of the system software.

Table 2-1: Elements of the Amiga Graphical User Interface System

| System Element | Purpose |
|-------------------|---|
| Intuition | The main toolkit and function library for creating a graphical user interface (GUI) on the Amiga. |
| Workbench | The Amiga file system GUI in which icons represent applications and files. |
| Preferences | A family of editors and configuration files for setting Amiga system options. |
| BOOPSI | Subsystem of Intuition that allows applications to add extensions to Intuition through object-oriented techniques (Release 2 only). |
| Gadtools Library | A support library for creating Intuition gadgets and menus (Release 2 only). |
| ASL Library | A support library for creating Intuition requesters (Release 2 only). |
| Icon Library | Main library for using Workbench icons. |
| Workbench Library | A support library for Workbench icons and menus (Release 2 only). |
| Console Device | An I/O support module which allows windows to be treated as text-based virtual terminals. |
| Graphics Library | The main library of rendering and drawing routines. |
| Layers Library | A support library that manages overlapping, rectangular drawing areas which Intuition uses for windows. |

As you read about Intuition in the chapters to follow, you will be introduced to each of these elements of the Amiga user interface in more detail.

1.4 2 / About User Interfaces / Goals of Intuition

Intuition was designed with two major goals in mind. The first is to give users a friendly and consistent environment to control the functions of the Amiga operating system and its applications.

The second goal (the big one) is to give application designers a graphical user interface toolkit that manages all the complexities of sharing the system with other programs that may be running at the same time. Since the Amiga is a multitasking computer, many programs can reside in memory at the same time sharing the system's resources with one another. Programs take turns running so that, from the user's point of view, it appears that many programs are running simultaneously.

On a multitasking computer like the Amiga, the user interface design must allow the user to control many programs with just one monitor, one keyboard, and one mouse. (Imagine driving many cars simultaneously with one steering wheel.) Intuition supplies the tools needed to solve this problem.

1.5 2 Intuition and the Graphical Interface / How the User Sees Intuition

Intuition solves the problem of interacting with multiple programs by dividing the display up into multiple screens and overlapping windows so that each application has its own work area. The user sees the Amiga environment through these windows, each of which can represent a different task or application context.

Figure 2-1: The Workbench Screen With Windows

The user performs operations inside screens and windows with the mouse, a mechanical device that moves a pointer over the Amiga's display. The user moves the mouse to position the pointer on graphic symbols of various objects or actions. Buttons on the mouse are pressed to select or activate the item pointed to.

The user can switch back and forth between different jobs, such as writing a document, drawing an illustration, printing text, or getting help from the system simply by moving from one window to another with the mouse. With the mouse, the user can also change the shape and size of application windows, move them around on the screen, overlap them, bring a window to the foreground, and send a window to the background. By changing the arrangement of the windows, the user selects which information is visible and which application to work with next. (Screens may also be moved up or down in the display, and they can be moved in front of or behind other screens.)

Workbench and Preferences Intuition's 3D Look

1.6 2 / How the User Sees Intuition / Workbench and Preferences

Normally, the Workbench screen (shown above) is the first screen the user sees upon booting the Amiga. Workbench is a special program supplied with every Amiga that gives the user a friendly and consistent graphic interface to the file system. It's the default environment the user starts out with.

In Workbench, disks, directories, files and other objects are symbolized by small pictures called icons which can be manipulated with the mouse. For instance, a program file can be executed by pointing to its icon with the mouse and double-clicking the left mouse button. The Workbench screen is automatically set up by Intuition and can be easily shared, so many application programs use it too.

User control of the OS is also supported through Preferences. Preferences is a family of editors and associated configuration files that allow the user to control the basic set up of the operating system. For example Printer Preferences sets up all the printer options.

Workbench, together with Preferences, gives the user an easy way to control the OS and launch applications. These programs are built with the same Intuition tools available to application programmers giving the whole Amiga system an integrated look and feel. Workbench and Preferences are important components of the Amiga graphic user interface system and are discussed in greater detail in later chapters.

1.7 2 / How the User Sees Intuition / Intuition's 3D Look

The Amiga operating system comes in different versions. The latest version, Release 2, contains significant improvements in the appearance of the Intuition graphical user interface, usually referred to as the 3D Look of Intuition.

Figure 2-2: An Example of the 3D Look of Intuition

In the new 3D look of Intuition, objects are drawn so that light appears to come from the upper left of the display with shadows cast to the lower right. Using light and shadow gives the illusion of depth so that images appear to stand out or recede from the display. By convention, an image with a raised appearance indicates an object that is available for use or modifiable. An image with a recessed appearance indicates an object that is unmodifiable, or for display purposes only. Applications should follow the same conventions.

Release 2 has other improvements over 1.3 (V34) and earlier versions of the operating system. Among these are new display resolutions, display sizes, and new function libraries to support Intuition. Most of the examples listed in this book assume Release 2. Where appropriate, the old 1.3 methods are also described.

1.8 2 Intuition & Graphical Interface / How an Application Sees Intuition

Intuition is organized as a library of over 100 functions. Before using an Intuition function you must first open the Intuition library. (In general, you must always open a library before you can call the functions of that library. See Chapter 1, "Introduction to Amiga System Libraries".)

- Components of Intuition
- Screens and Windows
- Gadgets, Menus and Requesters

1.9 2 / How an Application Sees Intuition / Components of Intuition

The types of data objects that the Intuition library functions create and control fall into six broad categories. These are the main components an application uses to build and operate a graphic user interface on the Amiga.

Table 2-2: GUI Components of Intuition

| | |
|--------------|--|
| Screens | The display environment. Sets the resolution and number of colors. |
| Windows | A graphic rectangle within a screen representing a working context. |
| Menus | A list of choices displayed at the top of a screen that can be selected with the mouse. |
| Gadgets | A control symbolized by a graphic image that can be operated with the mouse or keyboard. |
| Requesters | Sub-windows for confirming actions, accessing files and other special options. |
| Input events | Mouse, keyboard or other input activity. |

1.10 2 / How an Application Sees Intuition / Screens and Windows

As mentioned earlier, Intuition allows multiple programs to share the display by managing a system of multiple screens and overlapping windows. A screen sets up the display environment and forms the background that application windows operate in. A window is simply a graphic rectangle that represents a work context. Each screen can have many windows on it.

Multiple screens and windows give each application its own separate visual context so that many programs can output graphics and text to the display at the same time without interfering with one another. Intuition (using the layers library) handles all the details of clipping graphics so they stay inside window bounds and remembering graphics that go temporarily out

of sight when the user rearranges windows.

The keyboard and mouse are shared among applications through a simpler technique: only one application window at a time can have the input focus. Hence, Intuition ensures that only one window, called the active window gets to know about keyboard, mouse and other types of input activity.

Each application window is like a virtual terminal or console. Your program will seem to have the entire machine and display to itself. It can send text and graphics to its terminal window, and ask for input from any number of sources, ignoring the fact that other programs may be performing these same operations. Intuition handles all the housekeeping. In fact, your program can open several of these virtual terminals and treat each one as if it were the only program running on the machine. Intuition will keep track of all the activity and make sure commands and data are dispatched to the right place.

1.11 2 / How an Application Sees Intuition / Gadgets, Menus And Requesters

Intuition screens and windows provide an orderly way for multiple programs to share the display and input devices. Each application also needs a method for the user to send commands to it and select its options. Intuition supplies gadgets, menus and requesters for this purpose.

| | |
|---------|--------------------------------|
| Gadgets | Requesters |
| Menus | The Intuition Input Event Loop |

1.12 2 // Gadgets, Menus And Requesters / Gadgets

A gadget is an application control symbolized by a graphic image that can be operated with the mouse or keyboard. The imagery used for a gadget could look like a switch, a knob, a button, or just about anything. Intuition supplies some pre-fabricated gadgets, called system gadgets, for controlling window and screen arrangements. Other gadget types allow the user to select colors, enter text or numbers, and perform other simple operations.

Figure 2-3: An Intuition Window with Gadgets

Most of the user's input for a typical Intuition application will be obtained with gadgets. Gadgets are discussed in detail in Chapter 5, "Intuition Gadgets". Additional information on programming gadgets for Release 2 of the operating system can be found in Chapter 15, "GadTools Library".

1.13 2 // Gadgets, Menus And Requesters / Menus

Intuition also supplies a menu system for accepting commands and options from the user. A menu is a list of choices displayed at the top of the screen from which the user can select with the mouse. Each screen has one menu bar that all application windows operating on the screen share. Whichever window is active controls what appears in the menu bar.

Figure 2-4: An Intuition Menu

The current set of menu choices can always be brought into view by pressing the right mouse button (the menu button) thus providing the user with a familiar landmark even in unfamiliar applications. Menus allow the user to browse through the possible set of actions that can be performed giving an outline-like overview of the functions offered by a program.

Menus are discussed in detail in Chapter 6, "Intuition Menus". Additional information on programming menus for Release 2 of the operating system can be found in Chapter 15, "GadTools Library".

1.14 2 // Gadgets, Menus And Requesters / Requesters

Gadgets and menus do much of the work of getting commands and option choices from the user. Sometimes though, an application needs to get further information from a user in response to a command which has already been initiated. In that case, a requester can be used. A requester is a temporary sub-window, usually containing several gadgets, used to confirm actions, access files, or adjust the special options of a command the user has already given.

Figure 2-5: An Intuition Requester

Requesters are discussed in detail in Chapter 7, "Intuition Requesters and Alerts". Additional information on programming requesters for Release 2 of the system can be found in Chapter 16, "ASL Library".

1.15 2 // Gadgets, Menus And Requesters / The Intuition Input Event Loop

Once an application has set up the appropriate screen, window, gadgets menus and requesters, it waits for the user to do something. Intuition can notify an application whenever user activity occurs by sending a message. The message is simply a pointer to some memory owned by Intuition that contains an `IntuiMessage` data structure describing the user activity that occurred.

To wait for user activity or other events, the Exec library provides a special function named `Wait()`. The Exec `Wait()` function suspends your task allowing other applications or system tasks to run while your application is waiting for input or events from Intuition and other sources.

Thus, the basic outline for any Intuition program is:

- * Set up the window, screen and any required gadgets, menus or requesters.
- * Wait() for a message from Intuition about user activity or other events. Copy needed data from the message and tell Intuition you received it by replying. Look at the data and take the appropriate action.
- * Repeat until the user wants to quit.

These steps, sometimes referred to as the Intuition input event loop are basically the same for any Intuition application.

As you might expect, Intuition can send a message to your application whenever the user presses a key on the keyboard or moves the mouse. Other types of input events Intuition will notify you about include gadget hits, menu item selection, time elapsing, disk insertion, disk removal, and window rearrangement.

Gadgets, menus, requesters are the nuts and bolts of the Intuition GUI toolkit. Much of the code in an application that uses Intuition deals with the set up and operation of these important data objects. No matter how simple, complex, or fanciful your program design, it will fit within the basic Intuition framework of windows and screens, gadgets, menus and requesters. The users of the Amiga understand these basic Intuition elements and trust that the building blocks remain constant. This consistency ensures that a well-designed program will be understandable to the naive user as well as to the sophisticate.

1.16 2 Intuition and the Graphical Interface / A Simple Intuition Program

The sample Intuition program that follows shows all of the basic requirements for an Intuition application. There are three important points:

- * You must open the Intuition library before you can use the Intuition functions. Certain languages such as C require the pointer to the Intuition library to be assigned to a variable called IntuitionBase (see Chapter 1 for more about this).
- * When you set up a window, you also specify the events that you want to know about. If the user performs some activity that triggers one of the events you specified, Intuition signals you and sends a message. The message is a pointer to an IntuiMessage data structure that describes the event in more detail. Messages about Intuition events are sent to a MsgPort structure which queues up the messages for you in a linked list so that you may respond to them at your convenience.
- * Resources must be returned to the system. In this case, any windows, screens or libraries that were opened are closed before exiting.

Example Intuition Event Loop
Intuition Example (V36 And Later)

Intuition Example (All Versions)

1.17 2 / A Simple Intuition Program / Example Intuition Event Loop

The Intuition event loop used in the example is very simple. The example first sets up a custom screen, opens a window on it, then waits for Intuition to send messages about user input with the following event loop:

```
winsignal = 1L << window1->UserPort->mp_SigBit; /* window signal */
signalmask = winsignal; /* example only waits for window events */

while( !done ) {
    signals = Wait(signalmask);
    if (signals & winsignal)
        done = handleIDCMP(window1);
}
```

Intuition sends messages about user activity to a special port known as the IDCMP. Each window can have its own IDCMP (in the code above the IDCMP is window1->UserPort). To wait for event messages to arrive at the IDCMP port, the example code calls the Exec Wait() function. It then processes and replies to any event messages that it gets in a subroutine named handleIDCMP(). For this example, the only event Intuition will report is the close window event. When the example detects this event, it closes the window, closes the screen, closes the Intuition library and exits. Event loops similar to this one are used in Intuition examples throughout this book. For more information about IDCMP and user input, see the chapters on "Intuition Windows" and "Intuition Input and Output".