

GTEvents.mod

COLLABORATORS

	TITLE : GTEvents.mod		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		August 30, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	GTEvents.mod	1
1.1	GTEvents	1
1.2	Overview of GadTools event handling	2
1.3	Revision History	2
1.4	Global Switches	3
1.5	Imported Modules	3
1.6	Description of GadToolsPort	3
1.7	Using GadToolsPort	3
1.8	GadToolsPort Declarations	4
1.9	GadToolsPort Methods	4
1.10	HandleSig()	4
1.11	FlushPort()	5
1.12	Module Initialisation	5

Chapter 1

GTEvents.mod

1.1 GTEvents

```
< * STANDARD- * > < * MAIN- * >

MODULE GTEvents;
(*
    $RCSfile: GTEvents.mod $
    Description: Implementation of GadToolsPort class

    Created by: fjc (Frank Copeland)
    $Revision: 1.10 $
    $Author: fjc $
    $Date: 1995/01/26 00:46:19 $

    Copyright © 1994, Frank Copeland.
    This file is part of the Oberon-A Library.
    See Oberon-A.doc for conditions of use and distribution.
```

Module GTEvents extends the classes defined in Module Events to provide the event management required by the GadTools library introduced in Kickstart 2.0.

Overview Overview of GadTools event handling

Class descriptions

GadToolsPort Description of the GadToolsPort Class

Other

Switches	Global compiler switches
Imports	Imported modules
Initialisation	Module initialisation
Terminology	Definition of terms
History	Revision history

1.2 Overview of GadTools event handling

The GadTools library was introduced in Kickstart 2.0 to provide high level user-interface objects that previously had to be built by hand using Intuition's lower level facilities. Notification of events involving these objects is via the existing IDCMP message ports. However, GadTools objects are often complex, composite collections of related Gadgets, requiring a lot of work to manage. To spare programmers this task, events related to these Gadgets are processed and filtered by the GadTools library, which ensures that programmers only receive notification of events that are of interest to them.

To perform this filtering, GadTools library requires that programmers replace calls to Exec library's GetMsg() and ReplyMsg() functions with calls to GadTools' own GetIMsg() and ReplyIMsg() functions. These do the actual task of filtering events for GadTools objects, but pass through other events unchanged.

The GadToolsPort class is simply an extension of the IdcmpPort class which overrides the methods that rely on the Exec functions and replaces them with methods that use the GadTools functions.

1.3 Revision History

```
$Revision: 1.10 $
$Date: 1995/01/26 00:46:19 $
```

```
$Log: GTEvents.mod $
Revision 1.10  1995/01/26  00:46:19  fjc
- Release 1.5

Revision 1.9   1995/01/09  18:42:43  fjc
*** empty log message ***

Revision 1.8   1994/11/11  16:36:03  fjc
- Uses new external code interface.

Revision 1.7   1994/09/22  09:25:16  fjc
- Converted to new-style libcalls

Revision 1.6   1994/09/18  20:44:34  fjc
- Converted switches to pragmas/options

Revision 1.5   1994/09/03  16:25:29  fjc
*** empty log message ***

Revision 1.4   1994/08/08  16:19:59  fjc
Release 1.4

Revision 1.3   1994/06/14  02:06:07  fjc
- Updated for release

Revision 1.2   1994/06/04  15:50:17  fjc
```

- Changed to use new Amiga interface

Revision 1.1 1994/05/12 19:56:40 fjc

- Prepared for release

1.4 Global Switches

```
<*$ LongVars+ *> <*$ NilChk- *>
```

NIL checking is switched off and replaced with ASSERTs at the relevant points in the code.

1.5 Imported Modules

```
IMPORT
  SYS := SYSTEM, e := Exec, i := Intuition, gt := GadTools, ev := Events;
```

```
(*
  Exec          Exec library
  Intuition      Intuition library
  GadTools       GadTools library
  Events         Module Events
```

1.6 Description of GadToolsPort

A GadToolsPort is basically an IdcmpPort that has been modified to use the GadTools GetIMsg() and ReplyIMsg() functions.

Declarations	Constant and type declarations
Methods	GadToolsPort methods
Usage	Using GadToolsPort objects

1.7 Using GadToolsPort

A GadToolsPort is used in exactly the same way as an IdcmpPort. Simply replace any variables of type IdcmpPort with variables of type GadToolsPort. The only other requirement is to use ReplyIMsg() instead of ReplyMsg() in handler procedures that consume messages:

```
(*-----*)
PROCEDURE* HandleCloseWindow
  ( ip : IdcmpPort; msg : i.IntuiMessagePtr )
  : INTEGER;
```

```

BEGIN (* HandleCloseWindow *)
  gt.ReplyIMsg (msg);
  RETURN Stop
END HandleCloseWindow;

```

1.8 GadToolsPort Declarations

TYPE

```

GadToolsPort *= POINTER TO GadToolsPortRec;
GadToolsPortRec *= RECORD (ev.IdcmpPortRec) END;

```

(*

1.9 GadToolsPort Methods

OVERRIDDEN METHODS

Event handling

```

PROCEDURE ( gtp : GadToolsPort) HandleSig * () : INTEGER;
- Removes and processes messages queued at the object's MessagePort.

```

Message Port maintenance

```

PROCEDURE ( gtp : GadToolsPort ) FlushPort* ();
- Flushes any pending messages from the object's MessagePort.

```

1.10 HandleSig()

```

PROCEDURE (gtp : GadToolsPort) HandleSig * () : INTEGER;

```

(*

This procedure implements the behaviour required by receiving the signal associated with a MsgPort. The actual handling of the messages is delegated to gtp.HandleMsg().

*)

```

VAR result : INTEGER; msg : i.IntuiMessagePtr;

```

```

BEGIN (* HandleSig *)
  result := ev.Pass; (* Default *)

```

```

  (* Loop until all messages queued at the port are dealt with *)

```

```

  ASSERT (gtp.port # NIL, 132);

```

```

  LOOP

```

```

    (* Dequeue the next message, quit if there is none *)

```

```

    msg := gt.GetIMsg (gtp.port);

```

```

    IF msg = NIL THEN EXIT END;

```

```

    (* Despatch to the message handler *)

```

```

    result := gtp.HandleMsg (SYS.VAL (e.MessagePtr, msg));

    (* Process the return code *)
    IF result = ev.Pass THEN gt.ReplyIMsg (msg) END;
    IF result > ev.Continue THEN EXIT END
END;
RETURN result
END HandleSig;
(*

```

1.11 FlushPort()

```

PROCEDURE (gtp : GadToolsPort) FlushPort * ();
(*
    Gets and replies to all messages queued for the handler's message
    port. It is called inside an Exec.Forbid()/Exec.Permit() pair to
    prevent more messages from arriving at the port while it is being
    flushed.
*)

    VAR msg : i.IntuiMessagePtr;

BEGIN (* FlushPort *)
    ASSERT (gtp.port # NIL, 132);
    e.Forbid ();
    LOOP
        msg := gt.GetIMsg (gtp.port);
        IF msg = NIL THEN EXIT END;
        gt.ReplyIMsg (msg)
    END;
    e.Permit ()
END FlushPort;
(*

```

1.12 Module Initialisation

```

(* No initialisation required *)

END GTEvents.
(*

```