FDD_001_Enhanced Grid Navigation

Created with NTT Coding AI Tool. Please evaluate critically.

August 1, 2025

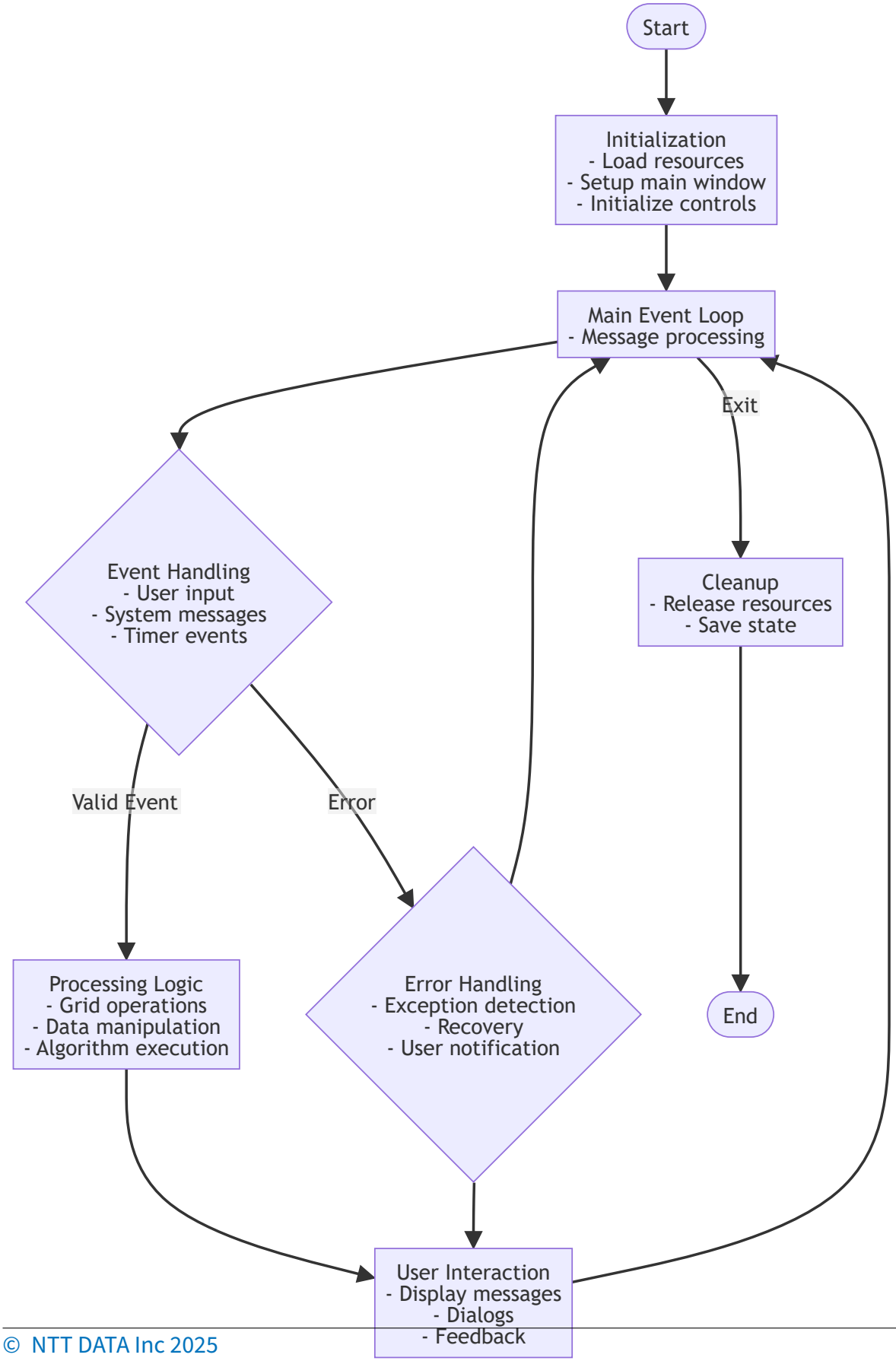
0.1 Purpose

This document provides a comprehensive functional specification for the MFC VC++ Grid Control Demo application, detailing its architecture, features, data flow, user interactions, and error management to support both manual testers and developers.

0.2 Overview

The Grid Control Demo is a Microsoft Foundation Classes (MFC) Visual C++ application that demonstrates a highly customizable grid control. The application allows users to interact with a grid, modify its structure and appearance, and test various cell types and behaviors. The project aims to showcase grid features such as editing, selection, drag-and-drop, clipboard operations, virtual mode, and advanced cell types, while providing robust error handling and user feedback.

0.3 Flow Diagram



0.4 Specific Functionalities

0.4.1 1. Grid Initialization and Configuration

- **Description:** Initializes the grid control, sets up default rows, columns, fixed rows/columns, and applies user-selected properties (editable, list mode, grid lines, etc.).
- **Exception Handling:** Catches memory exceptions during grid resizing and configuration, reporting errors to the user.

0.4.2 2. Dynamic Grid Modification

- **Description:** Allows users to add/delete rows and columns, change fixed rows/columns, and update grid structure in real time.
- **Exception Handling:** All modification operations are wrapped in TRY/CATCH blocks to handle memory allocation failures.

0.4.3 3. Cell Type Management

- **Description:** Supports various cell types (normal, read-only, checkbox, combo, URL, numeric, date/time) and allows users to set cell types dynamically.
- **Exception Handling:** Validates cell type changes and ensures type compatibility.

0.4.4 4. Editing and Selection

- **Description:** Enables in-place editing, single/multi-selection, and selection modes (row, column, cell). Editing can be programmatically rejected.
- **Exception Handling:** Editing attempts and changes can be programmatically rejected based on user settings.

0.4.5 5. Clipboard Operations

- **Description:** Supports copy, cut, and paste operations for grid data, with UI updates based on clipboard content.
- **Exception Handling:** Validates clipboard data format and availability.

0.4.6 6. Drag-and-Drop

- **Description:** Implements drag-and-drop for rows and columns, including OLE drag-and-drop support.
- **Exception Handling:** Ensures drag-and-drop is only enabled when the grid is properly registered and initialized.

0.4.7 7. Virtual Mode and Callback Functions

- **Description:** Supports virtual mode for large datasets, using callback functions for data retrieval and comparison.
- **Exception Handling:** Handles memory exceptions and ensures callback functions are valid.

0.4.8 8. Appearance Customization

- **Description:** Allows users to customize fonts, italics, title tips, grid lines, and vertical text orientation.
- **Exception Handling:** Validates font changes and UI updates.

0.4.9 9. Printing and Autosizing

- **Description:** Provides print functionality and automatic sizing of grid cells and columns.
- **Exception Handling:** Handles printing exceptions and ensures compatibility with device contexts.

0.4.10 10. Error Reporting and Tracing

- **Description:** Displays trace messages and error notifications in a dedicated trace window.
- **Exception Handling:** All error messages are formatted and displayed to the user.

0.5 Input/Output

Functionality	Input Data	Processing Logic	Output/Result
Grid Initialization	None (uses defaults or user settings)	Sets up grid structure, applies properties	Grid displayed with initial configuration
Add/Delete Row/Column	Row/column index, label	Inserts or deletes row/column, updates grid	Grid updated, UI refreshed
Set Cell Type	Cell coordinates, type identifier	Changes cell type, updates cell content	Cell displays new type
Edit Cell	Cell coordinates, new value	Validates and applies edit, or rejects based on settings	Cell updated or edit rejected
Clipboard Copy/Cut/Paste	Selected cells, clipboard data	Serializes/deserializes data, updates grid	Data copied, cut, or pasted

Functionality	Input Data	Processing Logic	Output/Result
Drag-and-Drop	Source/target cell/row/column	Moves or copies data, updates grid	Grid reflects drag-and-drop operation
Virtual Mode Data Retrieval	Row/column index	Calls callback to retrieve data	Cell displays callback data
Print Grid	Print command, device context	Formats and sends grid data to printer	Printed output
Trace/Error Reporting	Message string, error code	Formats and appends message to trace window	Trace window updated

0.6 Data Requirements

- **Grid Data:**
  - Rows, columns, fixed rows/columns (int)
  - Cell data: text (CString), image index (int), lParam (LPARAM), state (DWORD), format (DWORD), colors (COLORREF), font (LOGFONT*), margin (UINT)
- **Cell Types:**
  - Normal, read-only, checkbox, combo, URL, numeric, date/time (class type identifiers)
- **Clipboard Data:**
  - Text format (CF_TEXT)
- **Trace Messages:**
  - CString, displayed in a CEdit control
- **User Settings:**
  - BOOL flags for editable, list mode, grid lines, selection modes, etc.

0.7 Business Rules

- Grid must always have at least one row and one column.
- Editing can be programmatically rejected based on user settings.
- Only valid cell types can be assigned to a cell.
- Clipboard operations are only enabled when data is available and selection is non-empty.
- Drag-and-drop is only enabled when the grid is registered as a drop target.
- Virtual mode requires callback functions for data retrieval and comparison.
- UI must be updated after any change to grid structure or appearance.
- Error messages must be displayed to the user in the trace window.

0.8 User Interactions

- **Roles:**
  - End User: Interacts with the grid via the UI, modifies grid structure, edits cells, customizes appearance.
  - Developer/Tester: May use advanced features, test error handling, and extend functionality.
- **Permissions:**
  - All users have full access to grid features; no role-based restrictions.
- **Interaction Methods:**
  - Mouse: Click, double-click, right-click, drag-and-drop
  - Keyboard: Edit, navigate, select, copy/cut/paste
  - Menus/Buttons: Toggle features, customize appearance, print, clear trace

0.9 Detailed Steps

0.9.1 Example: Editing a Cell

1. User double-clicks a cell or presses Enter.
2. In-place edit control appears over the cell.
3. User types new value and presses Enter or clicks elsewhere.
4. Application validates the edit:
   - If `m_bRejectEditAttempts` is TRUE, edit is rejected immediately.
   - If `m_bRejectEditChanges` is TRUE, changes are discarded after editing.
5. If accepted, cell value is updated; otherwise, original value is restored.
6. Trace window logs the edit attempt and result.

0.9.2 Example: Adding a Row

1. User selects a row and clicks “Insert Row”.
2. Application inserts a new row at the selected position.
3. Grid is invalidated and redrawn.
4. Trace window logs the operation.

0.9.3 Example: Clipboard Copy

1. User selects one or more cells.
2. User presses Ctrl+C or selects “Copy” from the menu.
3. Application serializes selected data to clipboard in text format.
4. Trace window logs the copy operation.

0.10 Expected Outcomes

- Grid reflects all user-initiated changes immediately.
- All UI elements (menus, buttons) update to reflect current grid state.
- Trace window provides clear feedback for all operations and errors.
- Editing, selection, and clipboard operations behave as expected.
- Error conditions (e.g., memory allocation failure) are gracefully handled and reported.
- Printing produces a formatted, accurate representation of the grid.

0.11 Error Management

- **Memory Exceptions:**
  All grid modification operations (row/column changes, virtual mode) are wrapped in TRY/CATCH blocks. If a `CMemoryException` is caught, the error is reported to the user via a message box or the trace window.

```
TRY {
    m_Grid.SetRowCount(m_nRows);
}
CATCH (CMemoryException, e)
{
    e->ReportError();
    return;
}
END_CATCH
```

- **Edit Rejection:**
  Editing can be programmatically rejected by setting `*pResult = -1` in the edit notification handlers, based on user settings.
- **Clipboard Validation:**
  Clipboard operations check for data availability and correct format before enabling paste or processing data.
- **Drag-and-Drop Registration:**
  Drag-and-drop is only enabled if the grid is successfully registered as a drop target. Registration failures are handled gracefully.
- **UI Feedback:**
  All errors and important events are logged to the trace window for user awareness and debugging.
- **Resource Cleanup:**
  All dynamically allocated resources (fonts, memory, OLE objects) are properly released during cleanup and on application exit.

0.12 Potential Issues or Limitations

- **Performance in Virtual Mode:**
  Handling very large datasets (e.g., 100,000 rows) in virtual mode depends on the efficiency of callback functions. Poorly optimized callbacks may degrade performance.
- **Clipboard Format Support:**
  Only text format is supported for clipboard operations; rich data (e.g., formatting, images) is not preserved.
- **Drag-and-Drop Limitations:**
  Drag-and-drop is only supported within the grid; cross-application drag-and-drop is not implemented.
- **Font and Appearance Customization:**
  Some font or appearance changes may not be fully compatible with all cell types or may not render as expected on all systems.
- **Error Reporting:**
  Some low-level errors (e.g., GDI resource exhaustion) may not be fully recoverable and could require application restart.
- **Thread Safety:**
  The grid control is not thread-safe; all operations must be performed on the UI thread.
- **Platform Compatibility:**
  Some features may not be available or may behave differently on Windows CE or older Windows versions.
- **Extensibility:**
  Adding new cell types or behaviors requires extending the grid control and ensuring compatibility with existing features.

This document is intended to serve as a reference for both manual testers and developers, ensuring a clear understanding of the Grid Control Demo application's functionality, data flow, and error management strategies.