WHITE PAPER: ELIMINATING INTEGER ROUNDING ISSUES IN MULTI-OPERATION WINDOW RESIZING

INTRODUCTION

This document outlines a solution to address integer rounding inconsistencies observed in the Clarion Window Resize system during sequential resize operations. The current system demonstrates occasional dimension fluctuations, particularly with controls receiving partial scaling factors, due to the conversion between decimal calculations and integer-based Dialog Layout Units (DLUs).

THE PROBLEM

When a window undergoes multiple resize operations, control dimensions may exhibit unexpected behavior:

1. The resize system calculates positions using decimal scaling factors (e.g., 1.0193)
2. These calculations must be converted to integer DLUs
3. Each subsequent resize operation uses previously rounded values as its starting point
4. This leads to cumulative rounding errors or inconsistencies

For example, Button ID 8 demonstrated a 1 DLU width reduction during the second resize operation, where an increase would be expected.

PROPOSED SOLUTION: PERSISTENT RAW DIMENSION TRACKING

The solution involves maintaining a parallel set of non-rounded, floating-point dimensions for each control that persist across resize operations. This "Raw Dimension Queue" stores the mathematically accurate positions and sizes, applying rounding only at the final step of each operation.

IMPLEMENTATION DETAILS

1. Create a new queue to track raw dimensions for each control: RawDimensionQueue QUEUE,TYPE ControlID LONG RawXPos DECIMAL(8,4) RawYPos DECIMAL(8,4) RawWidth DECIMAL(8,4) RawHeight DECIMAL(8,4) OriginalWidth LONG OriginalHeight LONG ScaleCumulativeX DECIMAL(8,4) ScaleCumulativeY DECIMAL(8,4) END
2. Add this queue as a property to the LWindowResizeClass: LWindowResizeClass CLASS,TYPE (existing properties) RawDimensions &RawDimensionQueue END
3. Initialize the queue in the LWindowResizeClass.Construct method.
4. When each control is first added to the resize system, store its original dimensions in both integer and raw decimal form.
5. During resize operations, apply scaling factors to the raw dimensions rather than the current integer dimensions: Step 1: Retrieve raw dimensions for the control Step 2: Apply scaling factors to raw values

Step 3: Round to integers only for final display/positioning Step 4: Update the stored raw values for next resize operation

ALGORITHM OUTLINE

The key method would function as follows:

When scaling a control:

- Lookup control's entry in RawDimensions queue
- If not found, initialize with original dimensions
- Apply scaling to raw decimal values
- Convert to integers using consistent rounding technique
- Store updated raw values for subsequent operations
- Apply final integer dimensions to the control

IMPLEMENTATION CONSIDERATIONS

Data Structure Extensions

The LWindowResizeClass needs to be extended with:

- A pointer to the RawDimensions queue
- Methods to initialize and maintain raw dimension data
- Logic to use raw dimensions in scaling calculations

Memory Usage

The solution requires additional memory to store floating-point values for each control. For a window with n controls, this represents approximately:

- 28 bytes × n for the raw dimension values
- Negligible impact for typical windows with dozens of controls

Backward Compatibility

This enhancement maintains full backward compatibility:

- No changes to method signatures
- No changes to existing application code
- Improved resize behavior is transparent to calling code

TESTING METHODOLOGY

To validate this approach:

1. Implement the raw dimension tracking system
2. Create test windows with various control types
3. Perform multiple sequential resize operations
4. Compare control dimensions with predicted values
5. Verify consistent behavior across multiple resize cycles

PERFORMANCE IMPACT

The additional overhead is minimal:

- One additional queue lookup per control per resize operation
- Negligible calculation differences
- No impact on rendering or user interaction

ADDITIONAL BENEFITS

This approach offers several benefits beyond fixing the integer rounding issue:

1. Enables more predictable animations during resize operations
2. Provides foundation for more sophisticated mathematical transformations
3. Facilitates the implementation of proportional scaling systems
4. Supports future floating-point coordinate systems

INTEGRATION WITH BINDABLE INTERFACE

This enhancement provides a solid foundation for the planned BINDABLE interface:

- More consistent resize behavior improves predictability for script-driven adjustments
- Raw dimension data can be exposed through the interface if needed
- Mathematical consistency simplifies scripting logic

EXTENDED RAW DIMENSION QUEUE STRUCTURE

The RawDimensionQueue structure contains:

ControlID - Unique identifier for each control
RawXPos - Non-rounded X position as decimal
RawYPos - Non-rounded Y position as decimal
RawWidth - Non-rounded width as decimal
RawHeight - Non-rounded height as decimal
OriginalWidth - Initial integer width for reference
OriginalHeight - Initial integer height for reference

ScaleCumulativeX - Running product of all X scaling factors
ScaleCumulativeY - Running product of all Y scaling factors

This structure enables accurate tracking of dimensions through multiple resize operations.

ROUNDING STRATEGY

Proper rounding is crucial for this solution. The recommended approach is:

1. Store calculations with 4 decimal places precision
2. Use banker's rounding (round to nearest even) to minimize bias
3. Round only at the final stage before applying to controls
4. Apply minimum dimension constraints after rounding

MAINTAINING COORDINATE RELATIONSHIPS

This solution allows for better preservation of spatial relationships between controls:

1. Groups of controls can maintain proper alignment
2. Relative spacing remains more consistent
3. Edge-aligned controls stay properly aligned

CONCLUSION

The persistent raw dimension tracking approach eliminates inconsistencies in control resizing by maintaining mathematically accurate positions throughout multiple resize operations. This creates a more robust foundation for the resize system, ensuring predictable behavior regardless of how many resize operations occur.

This enhancement represents a minimal-impact change with significant benefits for resize behavior consistency, particularly for controls that receive partial scaling factors.

NEXT STEPS

1. Implement the RawDimensionQueue structure
2. Modify the SetPosition method to use raw dimensions
3. Add initialization logic to the Reset method
4. Test with various control types and window sizes
5. Integrate with planned BINDABLE interface development

APPENDIX: TESTING SCENARIOS

Test the solution with these specific scenarios:

1. Multiple consecutive small increases in window size
2. Multiple consecutive small decreases in window size
3. Alternating increases and decreases
4. Extreme size changes followed by returns to original size
5. Focus on controls with partial scaling factors (buttons, labels)
6. Test with nested container controls (sheets, regions, groups)
7. Test with tab controls and their special parent-child relationships