

View-Based NSTableView

Basic to advanced

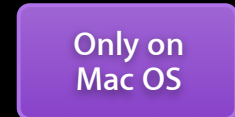
Session 120

Corbin Dunn

Cocoa Software Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Introduction



- NSTableView in Lion now supports NSView
- NSCell subclassing is no longer needed
- Easy design time layout
- Easy-to-animate changes
- Easy-to-customize drawing

What You Will Learn

- Layout
- Construction
- Bindings
- Customizing
- Drag and drop
- Animating

Note on “Cell”

- The term “cell” will be used to identify a view at a particular row/column
- It does not necessarily mean NSCell

Demo

TableViewPlayground

Available on <http://developer.apple.com/>

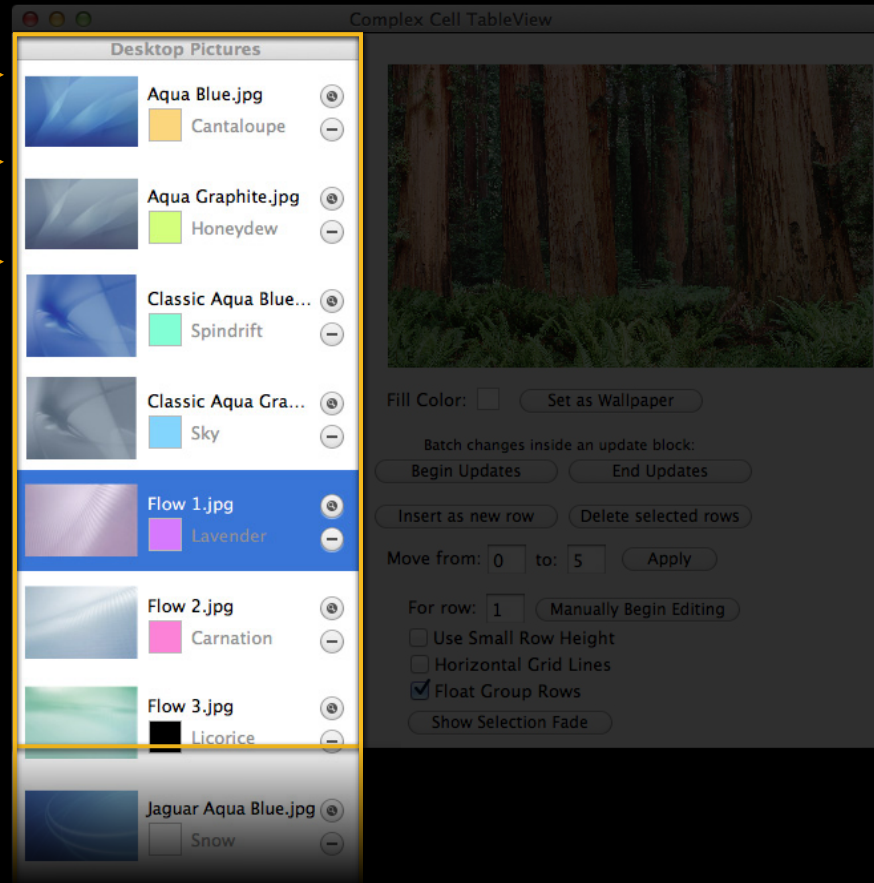
Layout

NSTableView Layout

NSScrollView

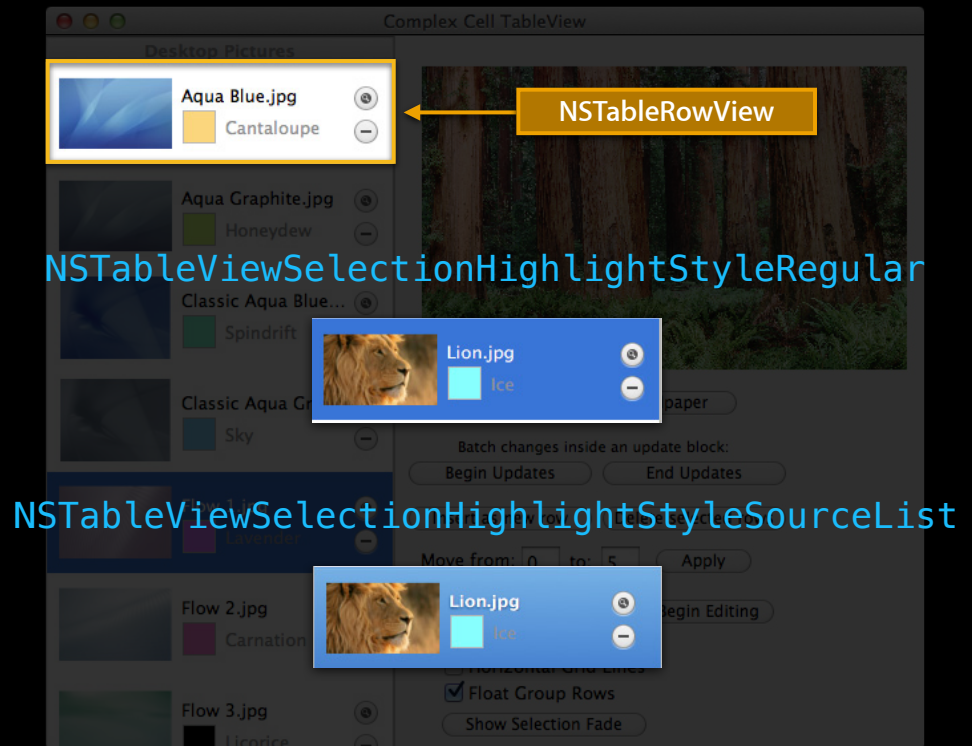
NSClipView

NSTableView



NSTableView Layout

- NSTableView draws
 - Selection



NSTableView RowView Layout

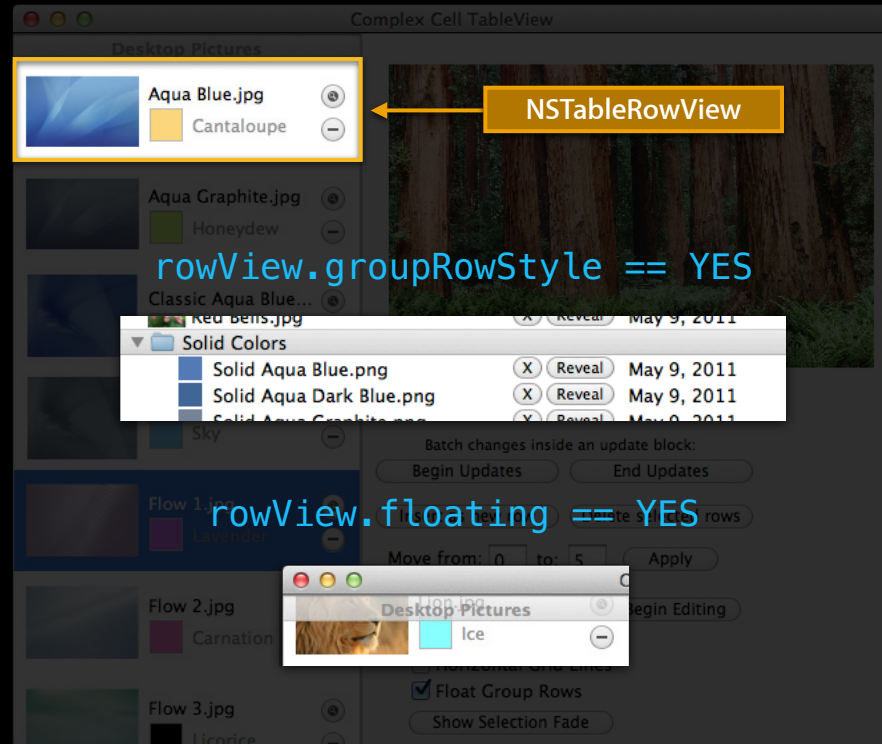
- NSTableView RowView draws
 - Row background color

`rowView.backgroundColor = [NSColor redColor];`

`rowView.backgroundColor = [NSColor greenColor];`

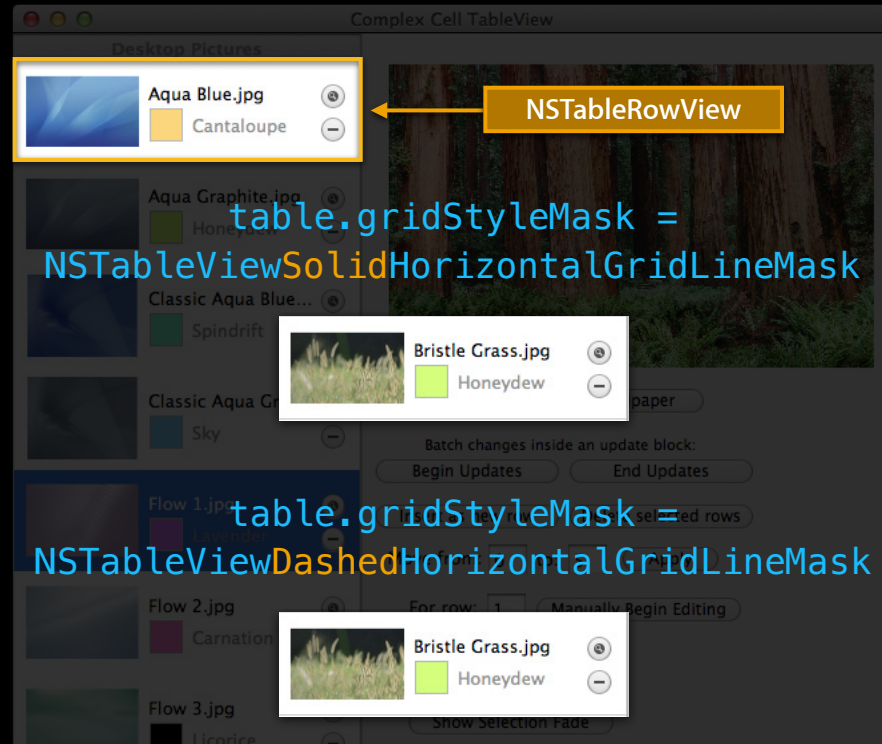
NSTableView RowView Layout

- NSTableView RowView draws
 - Group row style background
 - Set manually or via `-tableView:isGroupRow:`
 - Frequently used with floating
 - `floatsGroupRows = YES`



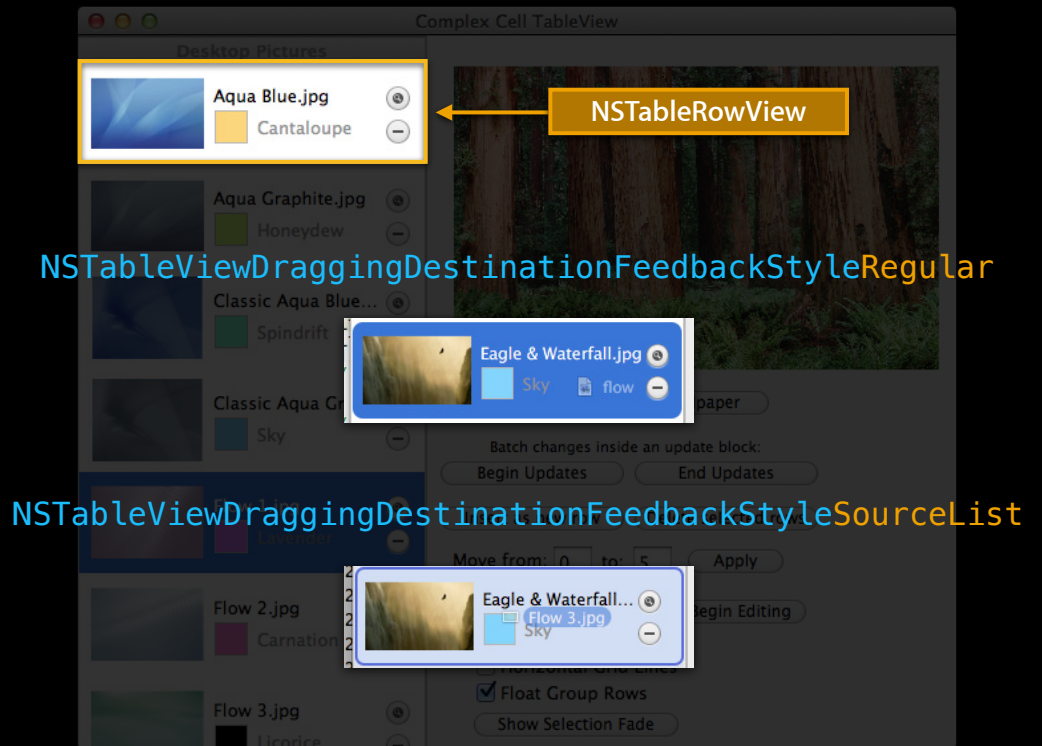
NSTableView RowView Layout

- NSTableView RowView draws
 - Bottom separators
 - Via NSTableView's gridStyleMask



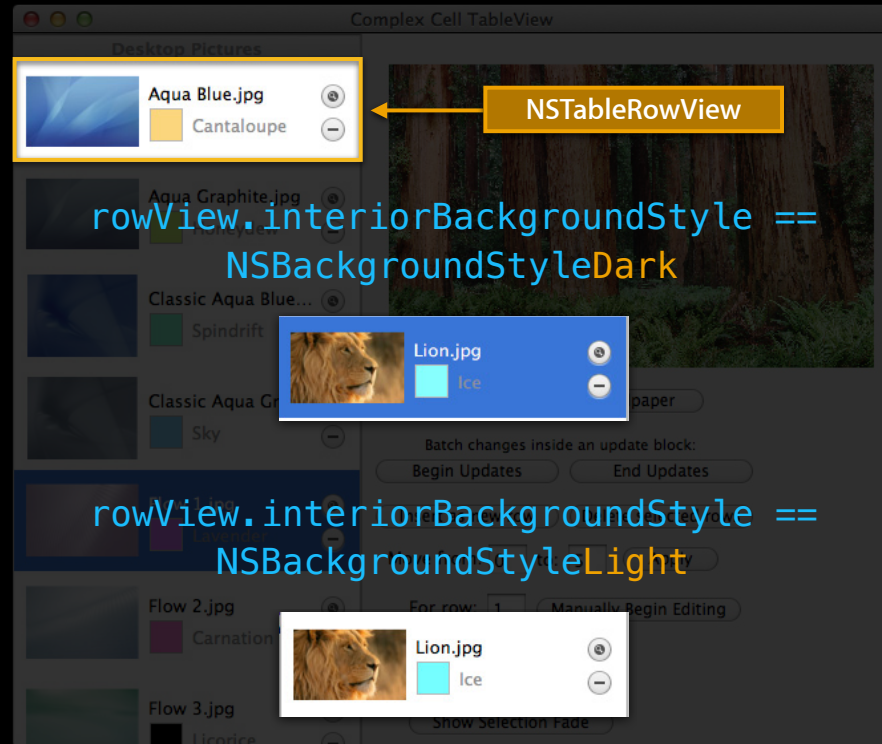
NSTableView Layout

- NSTableView draws
 - Drag-and-drop feedback



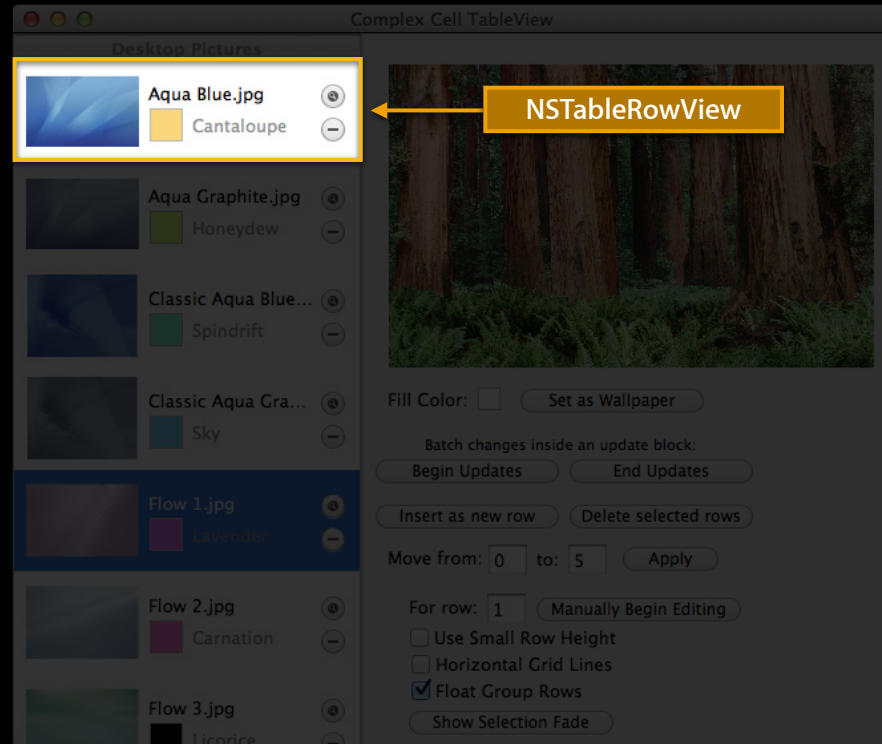
NSTableView RowView Layout

- NSTableView RowView computes
 - `-interiorBackgroundStyle` based on set properties



NSTableView RowView Layout

- NSTableView does not compute
 - All initial property values are set NSTableView
 - Opportunity to change the values in the delegate
 - tableView:didAddRowView:forRow:



NSTableColumn and NSTableRowView

- NSTableRowView stores
 - `numberOfColumns`
 - View per column
- Any NSView is allowed

The screenshot shows a table with three columns: Desktop Image, Date Modified, and Color. The table contains various entries, including images and colors. Annotations highlight different parts of the table structure:

- NSTableRowView**: A yellow box highlights a single row in the table, with an arrow pointing to the text label.
- NSTableColumn**: A yellow box highlights a single column in the table, with an arrow pointing to the text label.
- NSView with subviews**: A yellow box highlights the entire table area, with an arrow pointing to the text label.
- Regular NSTextField**: A yellow box highlights a text field in the 'Date Modified' column, with an arrow pointing to the text label.
- NSView with custom subviews**: A yellow box highlights a color swatch in the 'Color' column, with an arrow pointing to the text label.

NSTableView

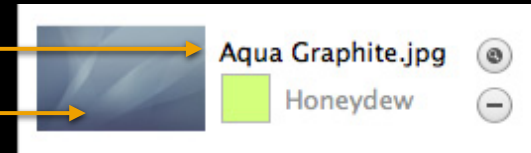
Optional, but allows easy hooks to subviews

```
@interface NSTableView : NSView ...
```

```
@property(assign) IBOutlet NSTextField *textField;
```

```
@property(assign) IBOutlet NSImageView *imageView;
```

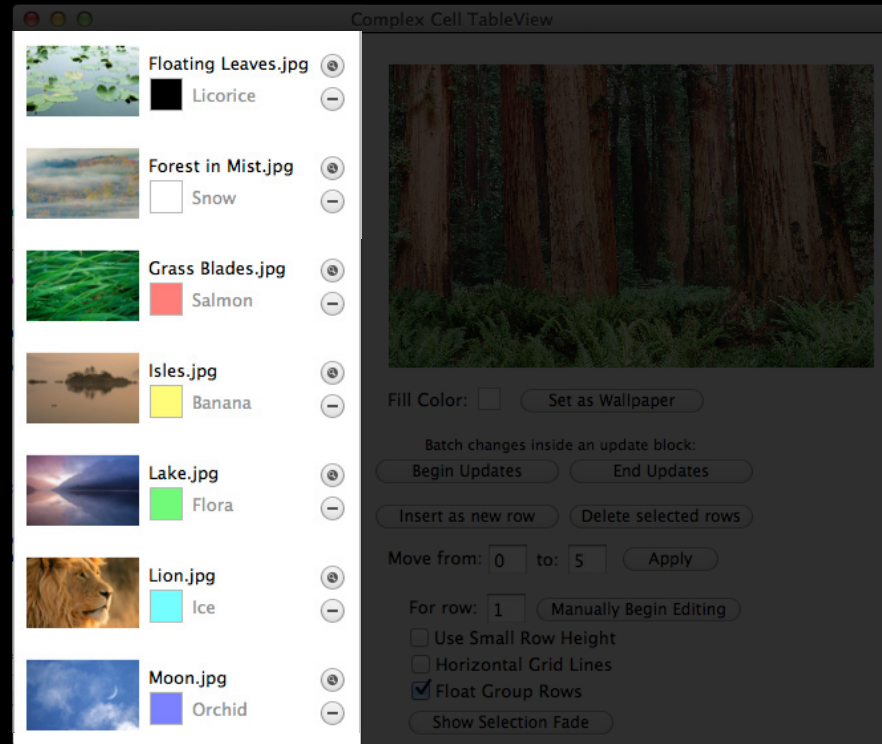
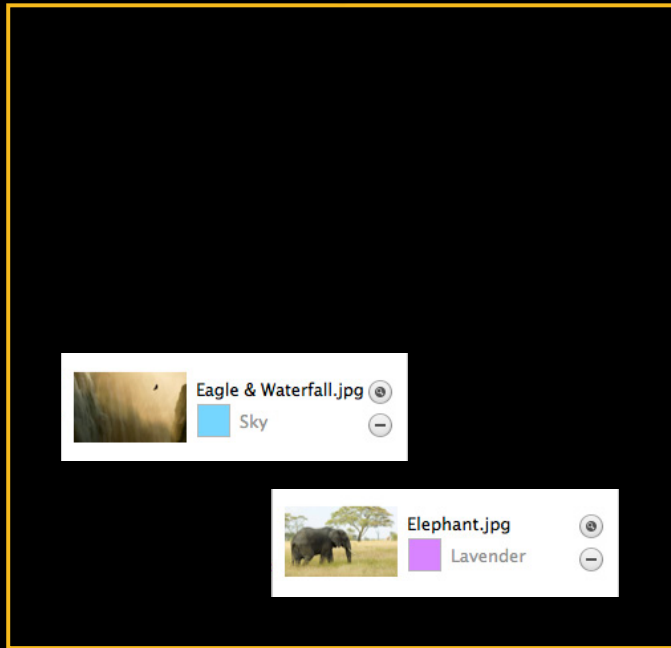
```
...
```



Also these are hints for accessibility

How Many Views?

Reuse Queue



Construction

NSTableView DataSource

Basic dataSource implementation

- Required method

```
- (NSInteger)numberOfRowsInTableView:(NSTableView *)tableView {  
    return array.count;  
}
```

- Optional method

```
- (id)tableView:(NSTableView *)tableView  
    objectValueForTableColumn:(NSTableColumn *)tableColumn  
    row:(NSInteger)row {  
    ...  
    return objectValue;  
}
```

NSTableView Delegate

Basic delegate implementation

- Required method

```
- (NSView *)tableView:(NSTableView *)tableView
    viewForTableColumn:(NSTableColumn *)tableColumn
        row:(NSInteger)row {

    NSTextField *result = [[NSTextField alloc] initWithFrame:...];
    result.stringValue = @"String value for row: %ld"; // Use 'row'
    return [result autorelease];
}
```

NSTableView Delegate

Basic delegate implementation

- Required method

```
- (NSView *)tableView:(NSTableView *)tableView  
  viewForTableColumn:(NSTableColumn *)tableColumn  
  row:(NSInteger)row {  
  
    NSTextField *result = [[NSTextField alloc] initWithFrame:...];  
    result.stringValue = @"String value for row: %ld"; // Use 'row'  
    return [result autorelease];  
}
```

NSTableView Delegate

Basic delegate implementation

- Required method

```
- (NSView *)tableView:(NSTableView *)tableView  
  viewForTableColumn:(NSTableColumn *)tableColumn  
  row:(NSInteger)row {  
  
  NSTextField *result = [[NSTextField alloc] initWithFrame:...];  
  result.stringValue = @"String value for row: %ld"; // Use 'row'  
  return [result autorelease];  
}
```

NSTableView Delegate

Basic delegate implementation

- Required method

```
- (NSView *)tableView:(NSTableView *)tableView  
  viewForTableColumn:(NSTableColumn *)tableColumn  
  row:(NSInteger)row {
```

```
    NSTextField *result = [[NSTextField alloc] initWithFrame:...];  
    result.stringValue = @"String value for row: %ld"; // Use 'row'  
    return [result autorelease];  
}
```

Optional if bindings are used

Using Identifiers

The screenshot shows a 'Complex OutlineView' window with three columns: 'Desktop Image', 'Date Modified', and 'Color'. Annotations with arrows point to specific cells:

- MainCell**: Points to the 'Desktop Image' column header.
- DateCell**: Points to the 'Date Modified' column header.
- ColorCell**: Points to the 'Color' column header.
- GroupCell**: Points to the 'Solid Colors' group header at the bottom of the list.

Desktop Image		Date Modified	Color
	X Reveal	May 9, 2011	Blueberry
	X Reveal	May 9, 2011	Magenta
	X Reveal	May 9, 2011	Iron
	X Reveal	May 9, 2011	Magnesium
	X Reveal	May 9, 2011	Mocha
	X Reveal	May 9, 2011	Fern
	X Reveal	May 9, 2011	Moss
	X Reveal	May 9, 2011	Ocean
	X Reveal	May 9, 2011	Eggplant
	X Reveal	May 9, 2011	Maroon
	X Reveal	May 9, 2011	Steel
	X Reveal	May 9, 2011	Aluminum
	X Reveal	May 9, 2011	Cayenne
	X Reveal	May 9, 2011	Asparagus
	X Reveal	May 9, 2011	Clover
	X Reveal	May 9, 2011	Teal
	X Reveal	May 9, 2011	Midnight
	X Reveal	May 9, 2011	Plum
▼ Solid Colors			
	X Reveal	May 9, 2011	Tin
	X Reveal	May 9, 2011	Nickel

NSView Identifier

NSView implements NSUserInterfaceItemIdentification

```
@protocol NSUserInterfaceItemIdentification
```

```
@property (copy) NSString *identifier;
```

```
@end
```

```
@interface NSView : NSResponder <NSUserInterfaceItemIdentification...> {
```

```
...
```

```
@end
```

NSTableView Delegate

More typical manual implementation

```
- (NSView *)tableView:(NSTableView *)tableView
    viewForTableColumn:(NSTableColumn *)tableColumn
        row:(NSInteger)row {

    NSTextField *result = [tableView makeViewWithIdentifier:@"MyView"
                                                                    owner:self];

    if (result == nil) {
        result = [[[NSTextField alloc] initWithFrame:...] autorelease];
        result.identifier = @"MyView";
    }
    result.stringValue = @"String value for row: %ld"; // Use 'row'
    return result;
}
```

NSTableView Delegate

More typical manual implementation

```
- (NSView *)tableView:(NSTableView *)tableView
    viewForTableColumn:(NSTableColumn *)tableColumn
        row:(NSInteger)row {

    NSTextField *result = [tableView makeViewWithIdentifier:@"MyView"
                                                                    owner:self];

    if (result == nil) {
        result = [[[NSTextField alloc] initWithFrame:...] autorelease];
        result.identifier = @"MyView";
    }
    result.stringValue = @"String value for row: %ld"; // Use 'row'
    return result;
}
```

NSTableView Delegate

More typical manual implementation

```
- (NSView *)tableView:(NSTableView *)tableView
  viewForTableColumn:(NSTableColumn *)tableColumn
  row:(NSInteger)row {

    NSTextField *result = [tableView makeViewWithIdentifier:@"MyView"
                                                                    owner:self];

    if (result == nil) {
        result = [[[NSTextField alloc] initWithFrame:...] autorelease];
        result.identifier = @"MyView";
    }
    result.stringValue = @"String value for row: %ld"; // Use 'row'
    return result;
}
```

NSTableView Delegate

Even more typical implementation with nibs

```
- (NSView *)tableView:(NSTableView *)tableView  
  viewForTableColumn:(NSTableColumn *)tableColumn  
    row:(NSInteger)row {  
  
    NSTextField *result = [tableView makeViewWithIdentifier:@"MyView"  
                                owner:self];  
  
    result.stringValue = @"String value for row: %ld"; // Use 'row'  
    return result;  
  
}
```

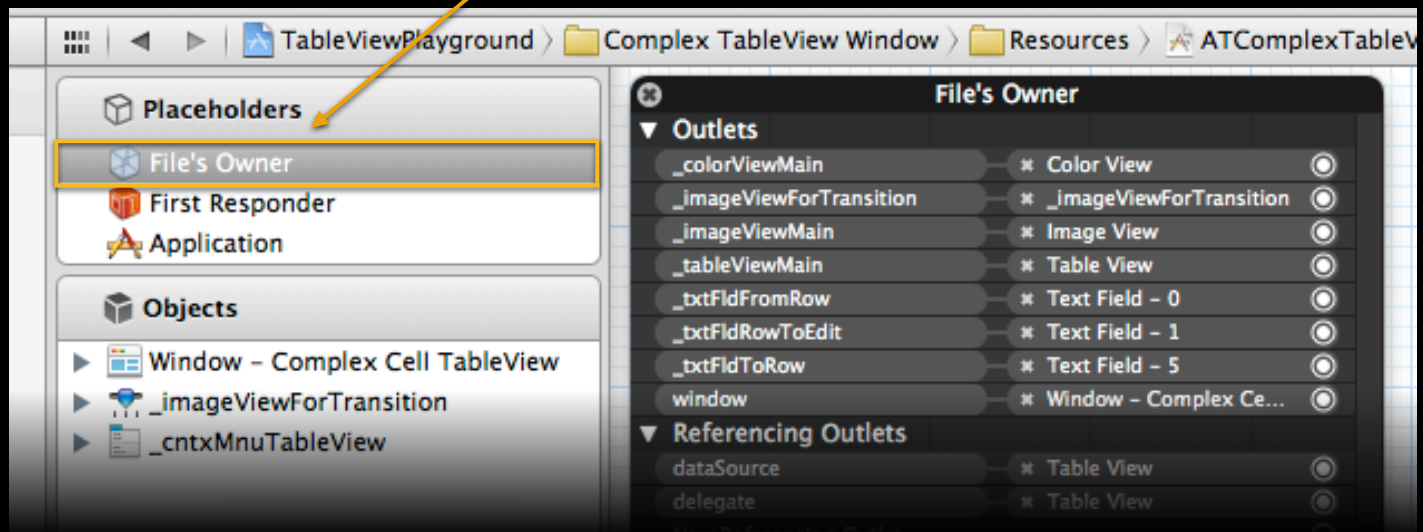
NSBundle Loading

Loading an NSView from an NSNib

```
@interface NSNib : NSObject <NSCoding> ...
```

```
- (BOOL)instantiateNibWithOwner:(id)owner  
    topLevelObjects:(NSArray **)topLevelObjects;
```

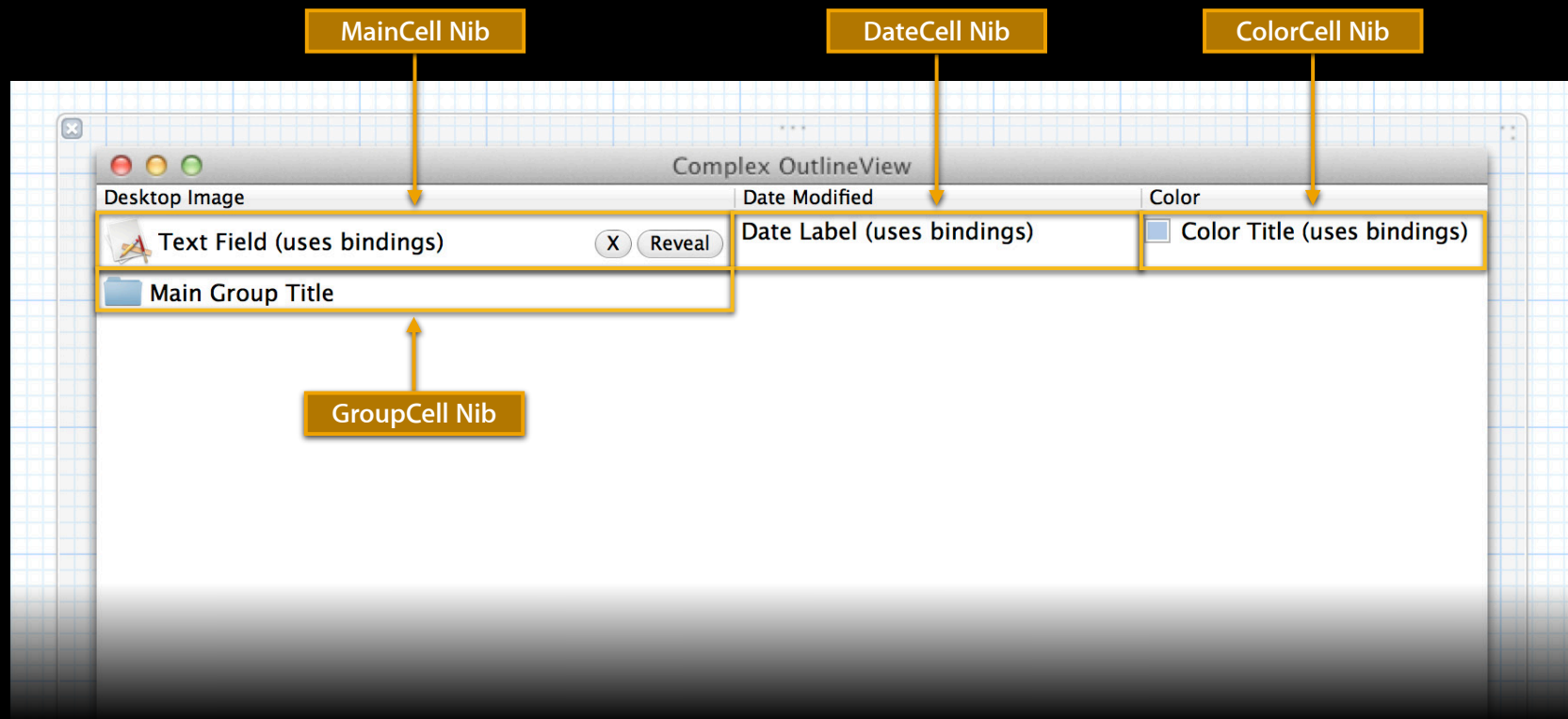
```
@end
```



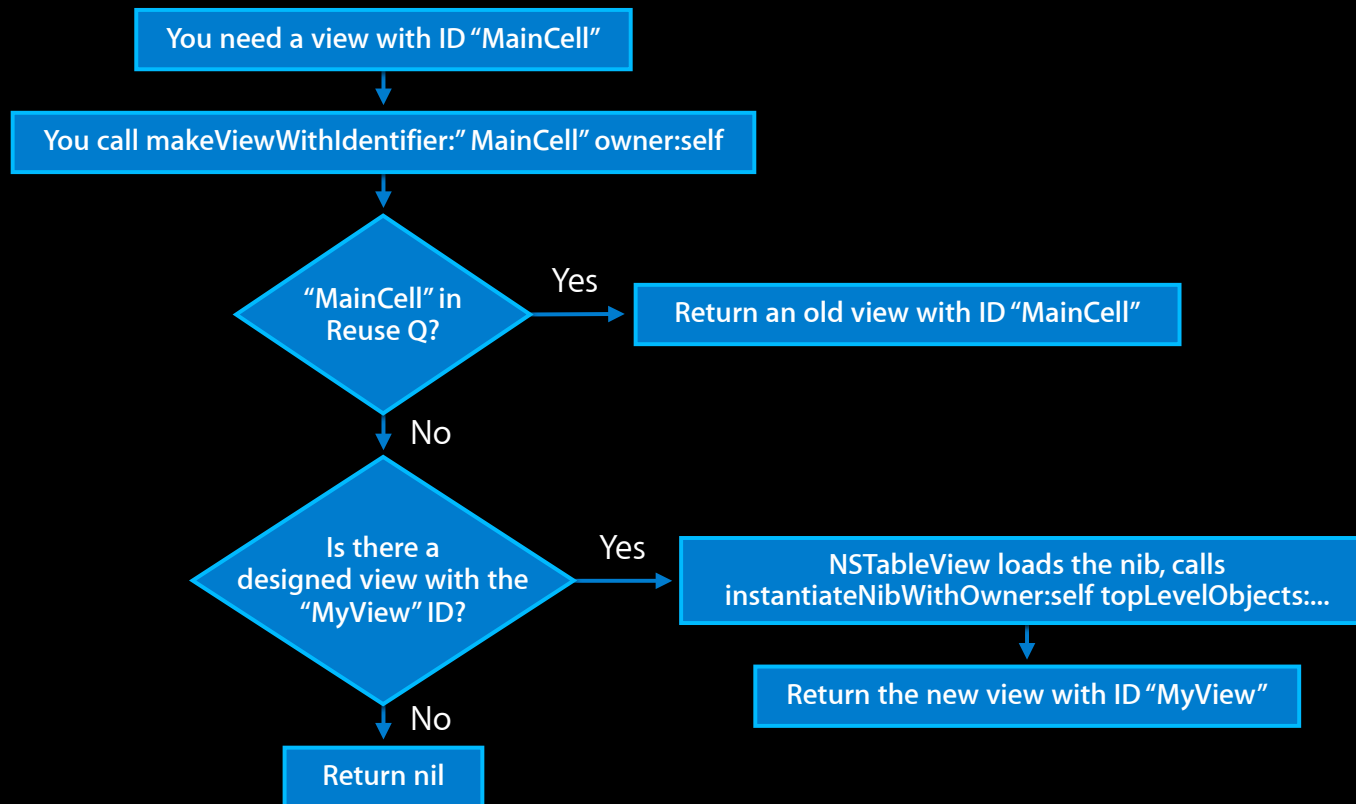
NSTableView Cell Encoding

Each cell is encoded as a separate NSNib

- Allows easy replication of the view and all bindings



makeViewWithIdentifier:owner:



Updating View State

- NSTableView updates NSTableRowView properties
- You must initialize your own properties

```
- (NSView *)tableView:(NSTableView *)tableView  
  viewForTableColumn:(NSTableColumn *)tableColumn  
  row:(NSInteger)row {
```

```
    NSTextField *result = [tableView makeViewWithIdentifier:@"MyView"  
                           owner:self];
```

```
    result.font = [NSFont ...]; // Reset the font!  
    result.stringValue = @"String value for row: %ld"; // Use 'row'  
    return result;
```

```
}
```

Demo

Basic Table Construction

BasicViewBasedTableView

Editing Completed Methods

- NSCell-based editing

```
- (void)tableView:(NSTableView *)tableView  
    setObjectValue:(id)object  
    forTableColumn:(NSTableColumn *)tableColumn  
        row:(NSInteger)row;
```

- NSView-based editing

- Target/action or notification

To Manually Begin Editing

- NSCell-based editing

```
- (void)editColumn:(NSInteger)column  
    row:(NSInteger)row  
    withEvent:(NSEvent *)theEvent  
    select:(BOOL)select;
```

- NSView-based editing

```
[window makeFirstResponder:textField];
```

How Editing Is Controlled

- NSTableView overrides -hitTest
- Calls a new NSResponder method with the hit view in question
 - (BOOL)validateProposedFirstResponder:(NSResponder *)responder
forEvent:(NSEvent *)event;
- NSTableView delays the first responder when text is hit
- Other controls can also call this method on the responder chain

Bindings

Binding Content

Providing data for a particular cell

- Normal NSArrayController bindings can be used
- The dataSource objectValue can also be used

```
- (id)tableView:(NSTableView *)tableView  
    objectValueForTableColumn:(NSTableColumn *)tableColumn  
        row:(NSInteger) row;
```

- Generally, return your model object value

Simple -objectValue

Desktop Image	Date Modified	Color
Beach.jpg	May 9, 2011	Blueberry
Bristle Grass.jpg	May 9, 2011	Magenta
Ducks on a Misty Pond.jpg	May 9, 2011	Iron
Eagle & Waterfall.jpg	May 9, 2011	Magnesium
Elephant.jpg	May 9, 2011	Mocha
Flamingos.jpg	May 9, 2011	Fern
Floating Leaves.jpg	May 9, 2011	Moss
Forest in Mist.jpg	May 9, 2011	Ocean
Grass Blades.jpg	May 9, 2011	Eggplant
Isles.jpg	May 9, 2011	Maroon
Lake.jpg	May 9, 2011	Steel
Lion.jpg	May 9, 2011	Aluminum
Moon.jpg	May 9, 2011	Cayenne
Mt. Fuji.jpg	May 9, 2011	Asparagus
Pink Forest.jpg	May 9, 2011	Clover
Pink Lotus Flower.jpg	May 9, 2011	Teal
Poppies.jpg	May 9, 2011	Midnight
Red Bells.jpg	May 9, 2011	Plum
Solid Colors		
Solid Aqua Blue.png	May 9, 2011	Tin
Solid Aqua Dark Blue.png	May 9, 2011	Nickel

```
@interface NSTextField : NSControl {...}
- (void)setObjectValue:(id<NSCopying>)obj;
...
@end
```

Identifier: NSDate

Class: NSTextField

Complex -objectValue

Desktop Image	Date Modified	Color
Beach.jpg	May 9, 2011	Blueberry
Bristle Grass.jpg	May 9, 2011	Magenta
Ducks on a Misty Pond.jpg	May 9, 2011	Iron
Eagle & Waterfall.jpg	May 9, 2011	Magnesium
Elephant.jpg	May 9, 2011	Mocha
Flamingo.jpg	May 9, 2011	Fern
Floating Leaves.jpg	May 9, 2011	Moss
Forest in Mist.jpg	May 9, 2011	Ocean
Grass Blades.jpg	May 9, 2011	Eggplant
Isles.jpg	May 9, 2011	Maroon
Lake.jpg	May 9, 2011	Steel
Lion.jpg	May 9, 2011	Aluminum
Moon.jpg	May 9, 2011	Cayenne
Mt. Fuji.jpg	May 9, 2011	Asparagus
Pink Forest.jpg	May 9, 2011	Clover
Pink Lotus Flower.jpg	May 9, 2011	Teal
Poppies.jpg	May 9, 2011	Midnight
Red Bells.jpg	May 9, 2011	Plum
Solid Colors		
Solid Aqua Blue.png	May 9, 2011	Tin
Solid Aqua Dark Blue.png	May 9, 2011	Nickel

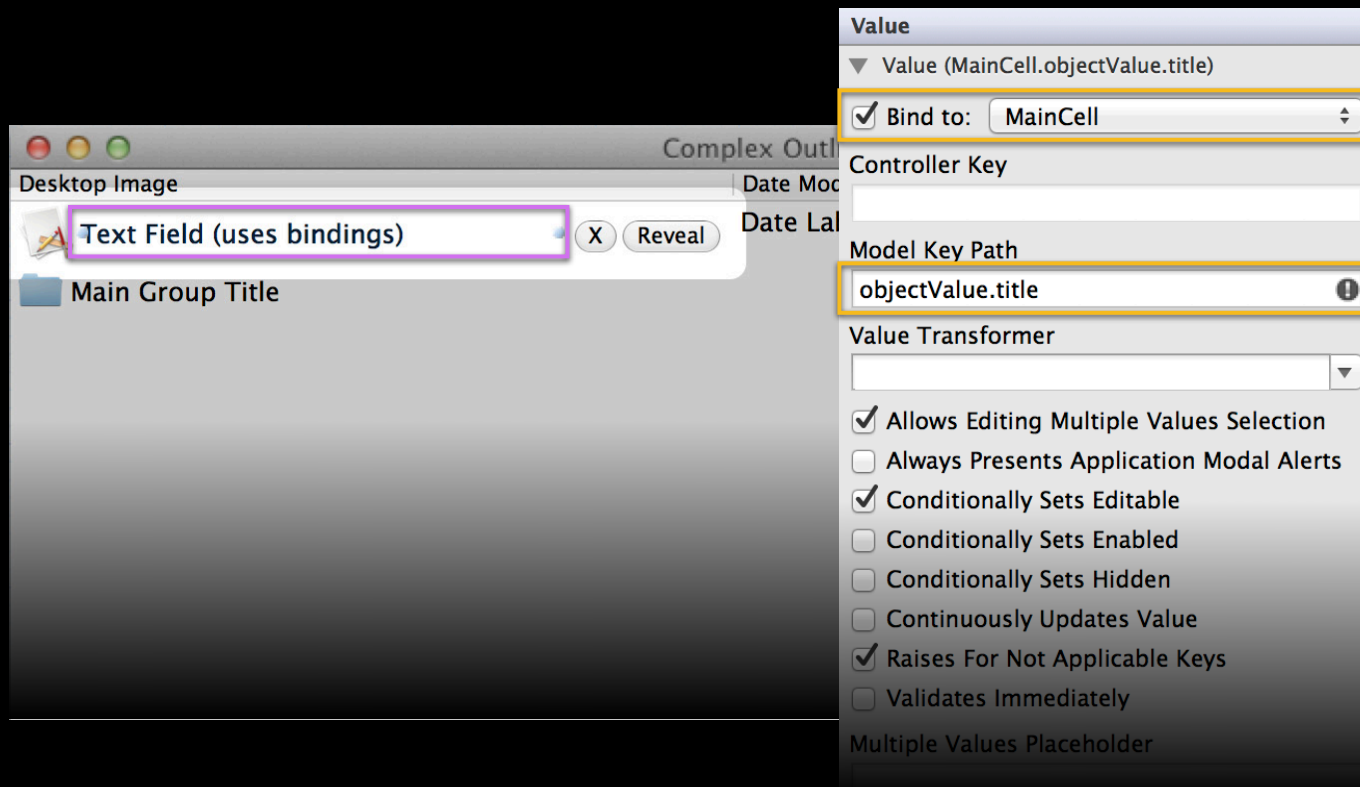
```
@interface NSTableView : NSView  
{ ... }  
  
@property(retain) id objectValue;
```

Complex -objectValue

Desktop Image		Date Modified	Color
	Bristle Grass.jpg	May 9, 2011	Blueberry
	Ducks on a Misty Pond.jpg	May 9, 2011	Iron
	Eagle & Waterfall.jpg	May 9, 2011	Magnesium
	Elephant.jpg	May 9, 2011	Mocha
	Flamingo.jpg	May 9, 2011	Fern
	Floating Leaves.jpg	May 9, 2011	Moss
	Forest in Mist.jpg	May 9, 2011	Ocean
	Isles.jpg	May 9, 2011	Eggplant
	Lake.jpg	May 9, 2011	Maroon
	Lion.jpg	May 9, 2011	Steel
	Moon.jpg	May 9, 2011	Aluminum
	Mt. Fuji.jpg	May 9, 2011	Cayenne
	Pink Forest.jpg	May 9, 2011	Asparagus
	Pink Lotus Flower.jpg	May 9, 2011	Clover
	Poppies.jpg	May 9, 2011	Teal
	Red Bells.jpg	May 9, 2011	Midnight
Solid Colors			
	Solid Aqua Blue.png	May 9, 2011	Tin
	Solid Aqua Dark Blue.png	May 9, 2011	Nickel

Inside Xcode 4

NSTableCellView has an objectValue binding



Automatic View Loading

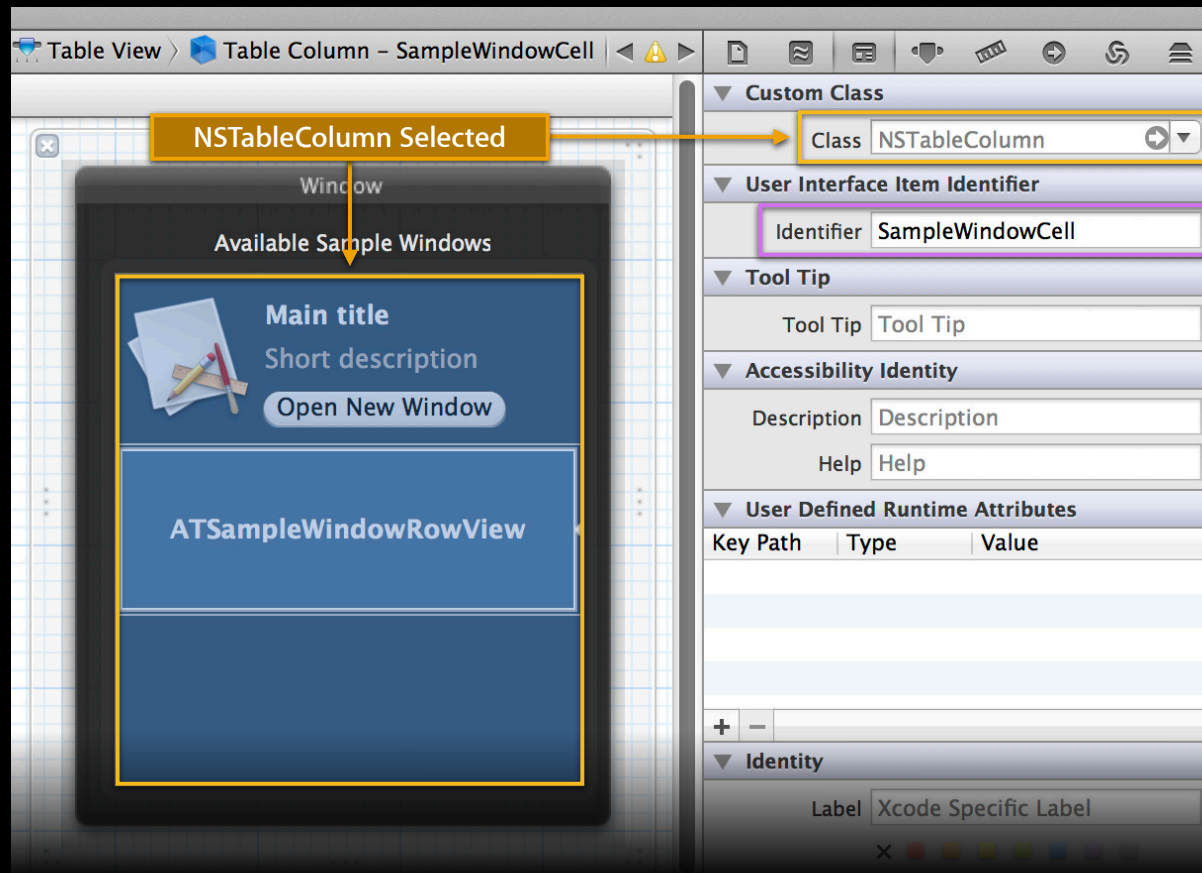
- Optional delegate method

`tableView:viewForTableColumn:row`

- NSTableColumn's identifier is used to look up a designed view

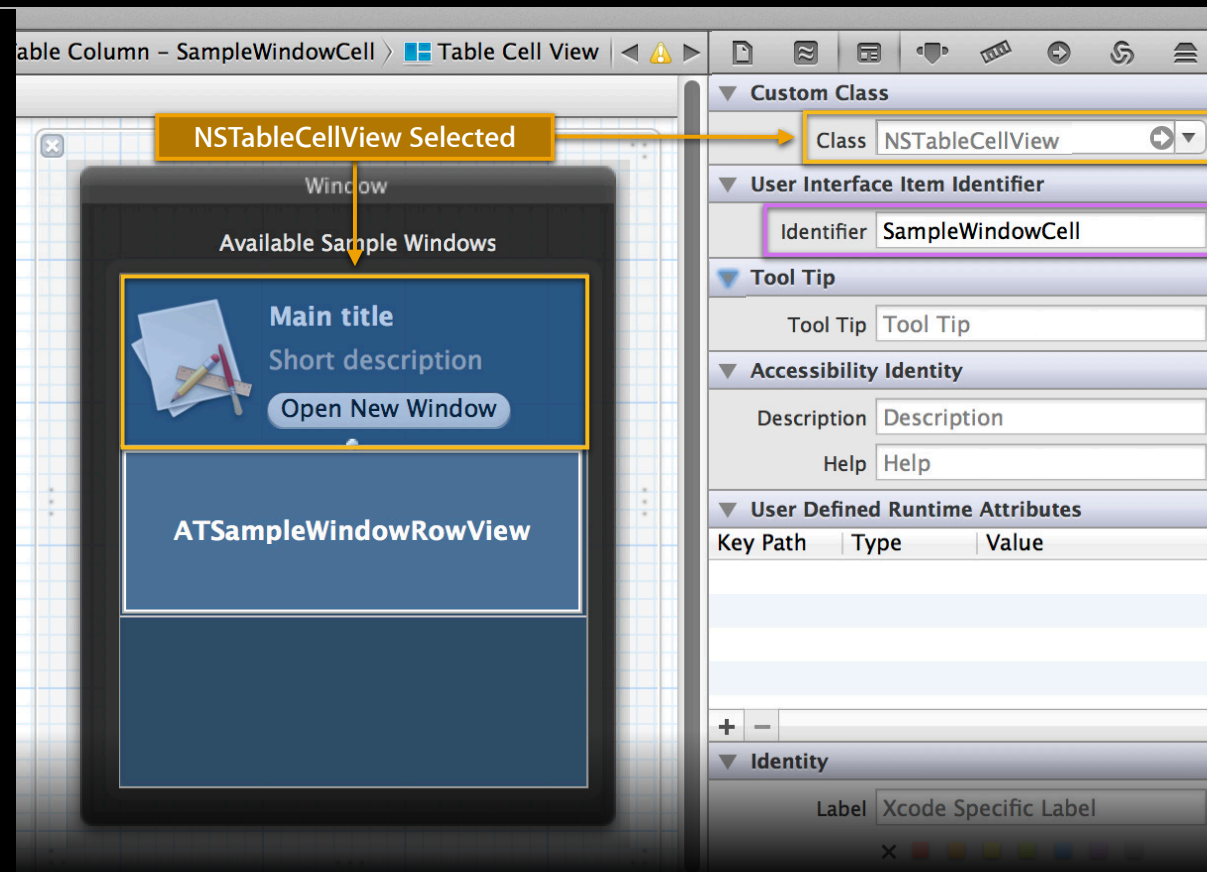
Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib



Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib



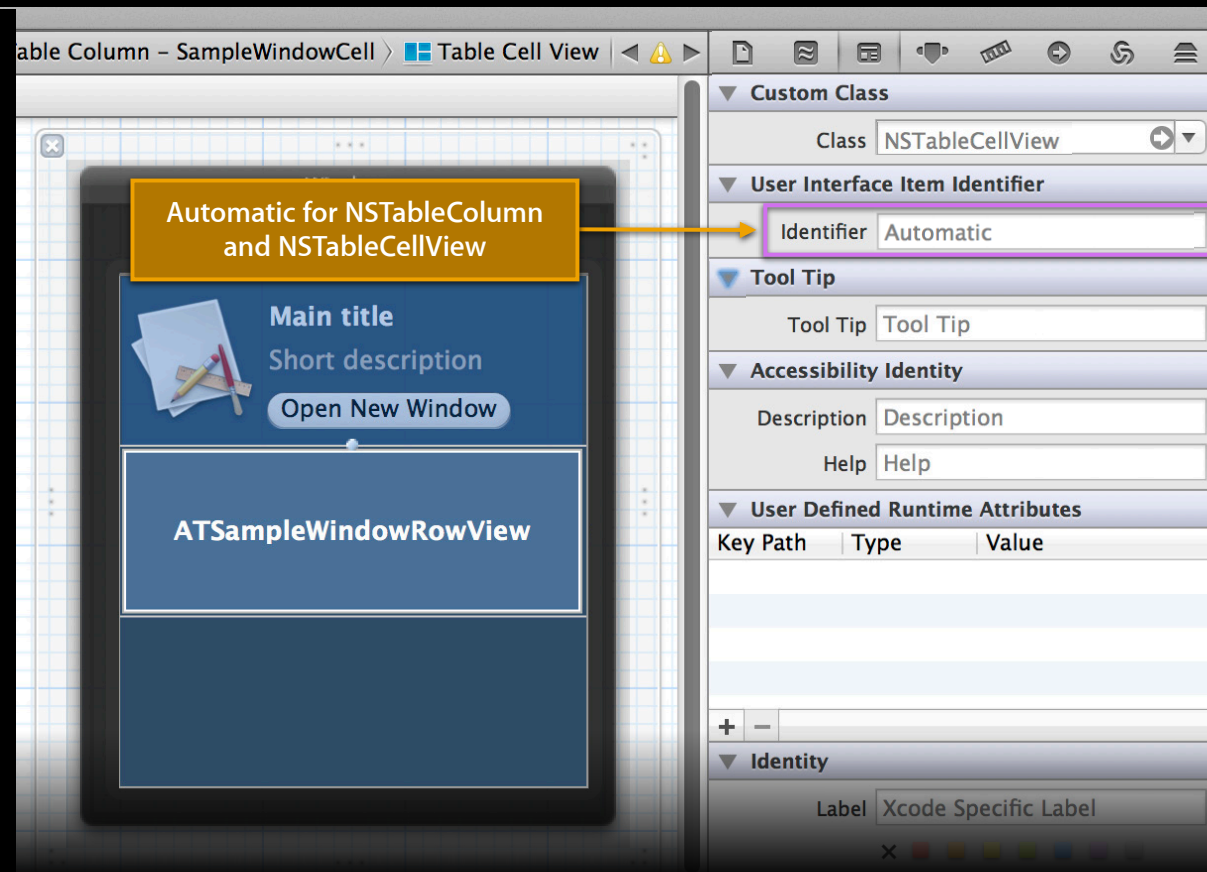
Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib

- Problem: Keeping the identifiers in sync
- Solution: Automatic identifiers in Xcode 4

Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib

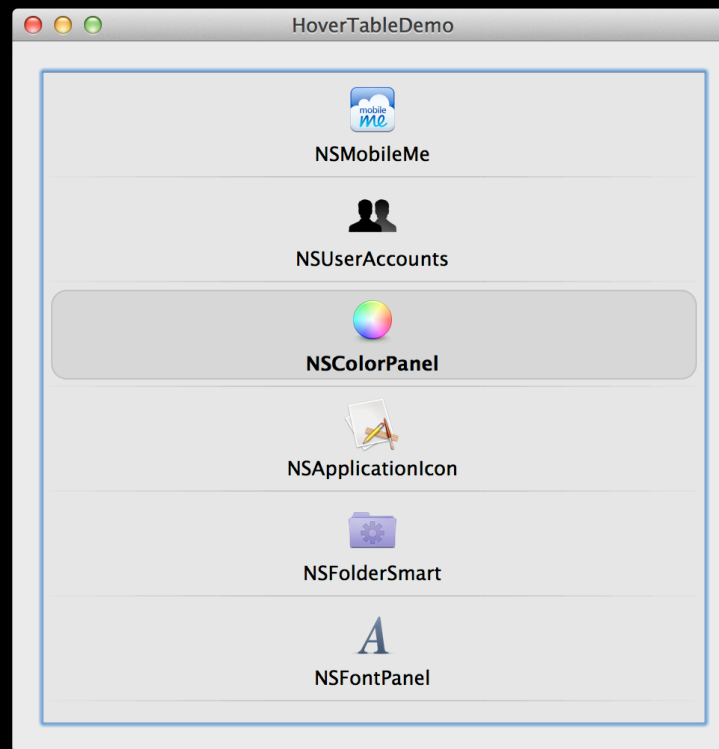


Demo

Customizing in the HoverTableDemo

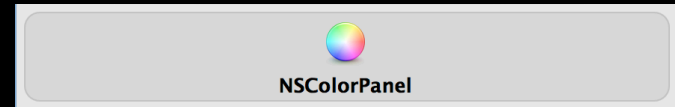
Custom Row Selection

Subclass NSTableView



Custom Row Selection

Subclass NSTableRowView

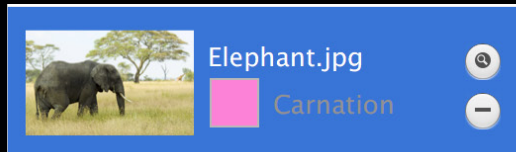


```
- (void)drawSelectionInRect:(NSRect)dirtyRect {
    NSRect selectionRect = NSInsetRect(self.bounds, 5.5, 5.5);
    [[NSColor colorWithCalibratedWhite:.72 alpha:1.0] setStroke];
    [[NSColor colorWithCalibratedWhite:.82 alpha:1.0] setFill];
    NSBezierPath *selectionPath = [NSBezierPath
        bezierPathWithRoundedRect:selectionRect xRadius:10 yRadius:10];
    [selectionPath fill];
    [selectionPath stroke];
}
```

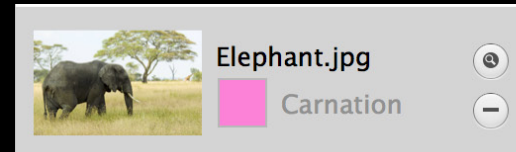
Custom Row Selection

Active selection or inactive selection

`rowView.emphasized == YES`

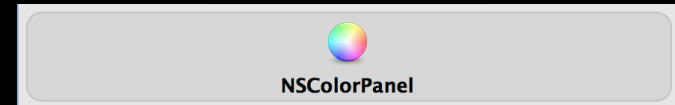


`rowView.emphasized == NO`



Bold Selected Text

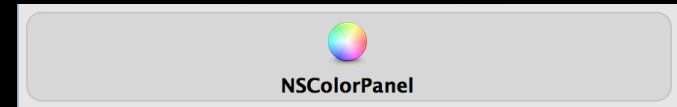
NSTableView delegate



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```

Bold Selected Text

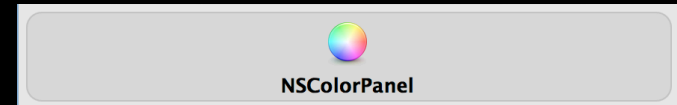
Bold text when selected



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```

Bold Selected Text

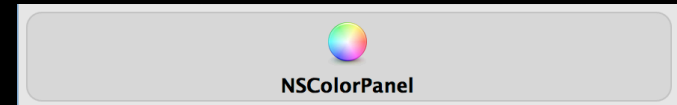
Bold text when selected



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```

Bold Selected Text

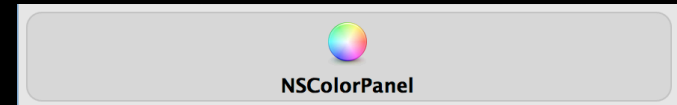
Bold text when selected



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```


Bold Selected Text

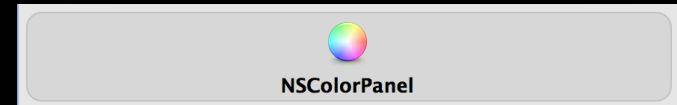
Bold text when selected



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```

Bold Selected Text

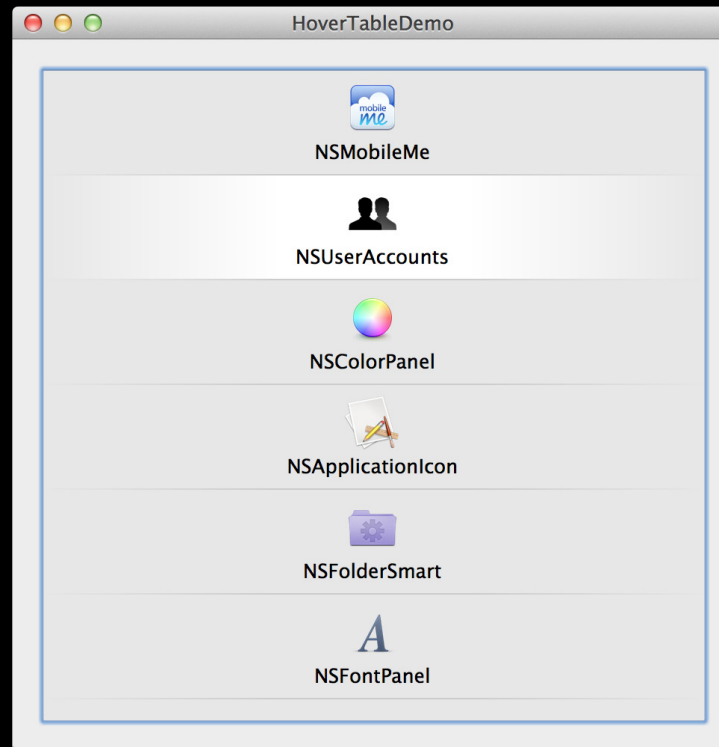
Bold text when selected



```
- (void)tableViewSelectionDidChange:(NSNotification *)notification {
    [_tableView enumerateAvailableRowViewsUsingBlock:
        ^(NSTableRowView *rowView, NSInteger row) {
            NSTableCellView *cellView = [rowView viewAtColumn:0];
            NSTextField *textField = cellView.textField;
            if (rowView.selected) {
                textField.font = [NSFont boldSystemFontOfSize:14];
            } else {
                textField.font = [NSFont systemFontOfSize:14];
            }
        }
    ];
}
```

Hover Effect

NSTableView subclass



Hover Effect

NSTableView subclass

```
- (void)updateTrackingAreas {
    [super updateTrackingAreas];
    if (_trackingArea == nil) {
        _trackingArea = [[NSTrackingArea alloc] initWithRect:NSZeroRect
                                                            options:... owner:self userInfo:nil];
    }
    if (![self trackingAreas] containsObject:_trackingArea) {
        [self addTrackingArea:_trackingArea];
    }
}
```



NSUserAccounts

Hover Effect

NSTableView subclass

```
- (void)mouseEntered:(NSEvent *)theEvent {
    _mouseInside = YES;
    [self setNeedsDisplay:YES];
}

- (void)mouseExited:(NSEvent *)theEvent {
    _mouseInside = NO;
    [self setNeedsDisplay:YES];
}
```



NSUserAccounts

Hover Effect

NSTableView subclass

```
- (void)drawBackgroundInRect:(NSRect)dirtyRect {
    [self.backgroundColor set];
    NSRectFill(self.bounds);

    if (_mouseInside) {
        NSGradient *gradient = [[[NSGradient alloc] init...] autorelease];
        [gradient drawInRect:self.bounds angle:0];
    }
}
```



NSUserAccounts

Hover Effect

NSTableView subclass

```
- (void)drawBackgroundInRect:(NSRect)dirtyRect {
    [self.backgroundColor set];
    NSRectFill(self.bounds);

    if (_mouseInside) {
        NSGradient *gradient = [[[NSGradient alloc] init...] autorelease];
        [gradient drawInRect:self.bounds angle:0];
    }
}
```



NSUserAccounts

Hover Effect

NSTableView subclass

```
- (void)drawBackgroundInRect:(NSRect)dirtyRect {
    [self.backgroundColor set];
    NSRectFill(self.bounds);

    if (_mouseInside) {
        NSGradient *gradient = [[NSGradient alloc] init...] autorelease];
        [gradient drawInRect:self.bounds angle:0];
    }
}
```



NSUserAccounts

Hover Effect

NSTableView subclass

```
- (void)drawBackgroundInRect:(NSRect)dirtyRect {
    [self.backgroundColor set];
    NSRectFill(self.bounds);

    if (_mouseInside) {
        NSGradient *gradient = [[NSGradient alloc] init...] autorelease];
        [gradient drawInRect:self.bounds angle:0];
    }
}
```



NSUserAccounts

Custom Separator

NSTableView subclass



NSMobileMe

```
- (void)drawSeparatorInRect:(NSRect)dirtyRect {
    NSRect separatorRect = self.bounds;
    separatorRect.origin.y = NSMaxY(separatorRect) - 1;
    separatorRect.size.height = 1;
    // Fill separatorRect with a gradient
    DrawSeparatorInRect(separatorRect);
}
```

Custom Separator

NSTableView subclass



NSMobileMe

```
- (void)drawSeparatorInRect:(NSRect)dirtyRect {
    NSRect separatorRect = self.bounds;
    separatorRect.origin.y = NSMaxY(separatorRect) - 1;
    separatorRect.size.height = 1;
    // Fill separatorRect with a gradient
    DrawSeparatorInRect(separatorRect);
}
```

Custom Separator

NSTableView subclass



NSMobileMe

```
- (void)drawSeparatorInRect:(NSRect)dirtyRect {
    NSRect separatorRect = self.bounds;
    separatorRect.origin.y = NSMaxY(separatorRect) - 1;
    separatorRect.size.height = 1;
    // Fill separatorRect with a gradient
    DrawSeparatorInRect(separatorRect);
}
```

Custom Separator

Dealing with empty rows



Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {
    // Only draw the grid past the last visible row
    NSInteger numberOfRows = self.numberOfRows;
    CGFloat yStart = 0;
    if (numberOfRows > 0) {
        yStart = NSMaxY([self rectOfRow:numberOfRows - 1]);
    }
    // Draw the first separator one row past the last row
    yStart += self.rowHeight;

    NSRect boundsToDraw = self.bounds;
    NSRect separatorRect = boundsToDraw;
    separatorRect.size.height = 1;
    while (yStart < NSMaxY(boundsToDraw)) {
```

Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {
    // Only draw the grid past the last visible row
    NSInteger numberOfRows = self.numberOfRows;
    CGFloat yStart = 0;
    if (numberOfRows > 0) {
        yStart = NSMaxY([self rectOfRow:numberOfRows - 1]);
    }
    // Draw the first separator one row past the last row
    yStart += self.rowHeight;

    NSRect boundsToDraw = self.bounds;
    NSRect separatorRect = boundsToDraw;
    separatorRect.size.height = 1;
    while (yStart < NSMaxY(boundsToDraw)) {
```

Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {
    // Only draw the grid past the last visible row
    NSInteger numberOfRows = self.numberOfRows;
    CGFloat yStart = 0;
    if (numberOfRows > 0) {
        yStart = NSMaxY([self rectOfRow:numberOfRows - 1]);
    }
    // Draw the first separator one row past the last row
    yStart += self.rowHeight;

    NSRect boundsToDraw = self.bounds;
    NSRect separatorRect = boundsToDraw;
    separatorRect.size.height = 1;
    while (yStart < NSMaxY(boundsToDraw)) {
```


Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {  
    ...  
    NSRect boundsToDraw = self.bounds;  
    NSRect separatorRect = boundsToDraw;  
    separatorRect.size.height = 1;  
    while (yStart < NSMaxY(boundsToDraw)) {  
        separatorRect.origin.y = yStart;  
        DrawSeparatorInRect(separatorRect); // Shared method  
        yStart += self.rowHeight;  
    }  
}
```

Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {  
    ...  
    NSRect boundsToDraw = self.bounds;  
    NSRect separatorRect = boundsToDraw;  
    separatorRect.size.height = 1;  
    while (yStart < NSMaxY(boundsToDraw)) {  
        separatorRect.origin.y = yStart;  
        DrawSeparatorInRect(separatorRect); // Shared method  
        yStart += self.rowHeight;  
    }  
}
```

Custom Separator

NSTableView subclass

```
- (void)drawGridInClipRect:(NSRect)clipRect {  
    ...  
    NSRect boundsToDraw = self.bounds;  
    NSRect separatorRect = boundsToDraw;  
    separatorRect.size.height = 1;  
    while (yStart < NSMaxY(boundsToDraw)) {  
        separatorRect.origin.y = yStart;  
        DrawSeparatorInRect(separatorRect); // Shared method  
        yStart += self.rowHeight;  
    }  
}
```

Providing Custom Row Views

Typical manual implementation

```
- (NSTableRowView *)tableView:(NSTableView *)tableView
    rowViewForRow:(NSInteger)row {
    MyRowView *result =
        [[MyRowView alloc] initWithFrame:NSMakeRange(0, 0, 100, 100)];
    // Setup custom properties on the row view
    return [result autorelease];
}
```

Providing Custom Row Views

Typical manual implementation

```
- (NSTableView *)tableView:(NSTableView *)tableView
    tableViewForRow:(NSInteger)row {
    MyRowView *result =
        [[MyRowView alloc] initWithFrame:NSMakeRange(0, 0, 100, 100)];
    // Setup custom properties on the row view
    return [result autorelease];
}
```

Providing Custom Row Views

Typical manual implementation

```
- (NSTableView *)tableView:(NSTableView *)tableView
    tableViewForRow:(NSInteger)row {
    MyRowView *result =
        [[MyRowView alloc] initWithFrame:NSMakeRange(0, 0, 100, 100)];
    // Setup custom properties on the row view
    return [result autorelease];
}
```

Providing Custom Row Views

Typical manual implementation

```
- (NSTableRowView *)tableView:(NSTableView *)tableView
    rowViewForRow:(NSInteger)row {
    MyRowView *result =
        [[MyRowView alloc] initWithFrame:NSMakeRange(0, 0, 100, 100)];
    // Setup custom properties on the row view
    return [result autorelease];
}
```

Providing Custom Row Views

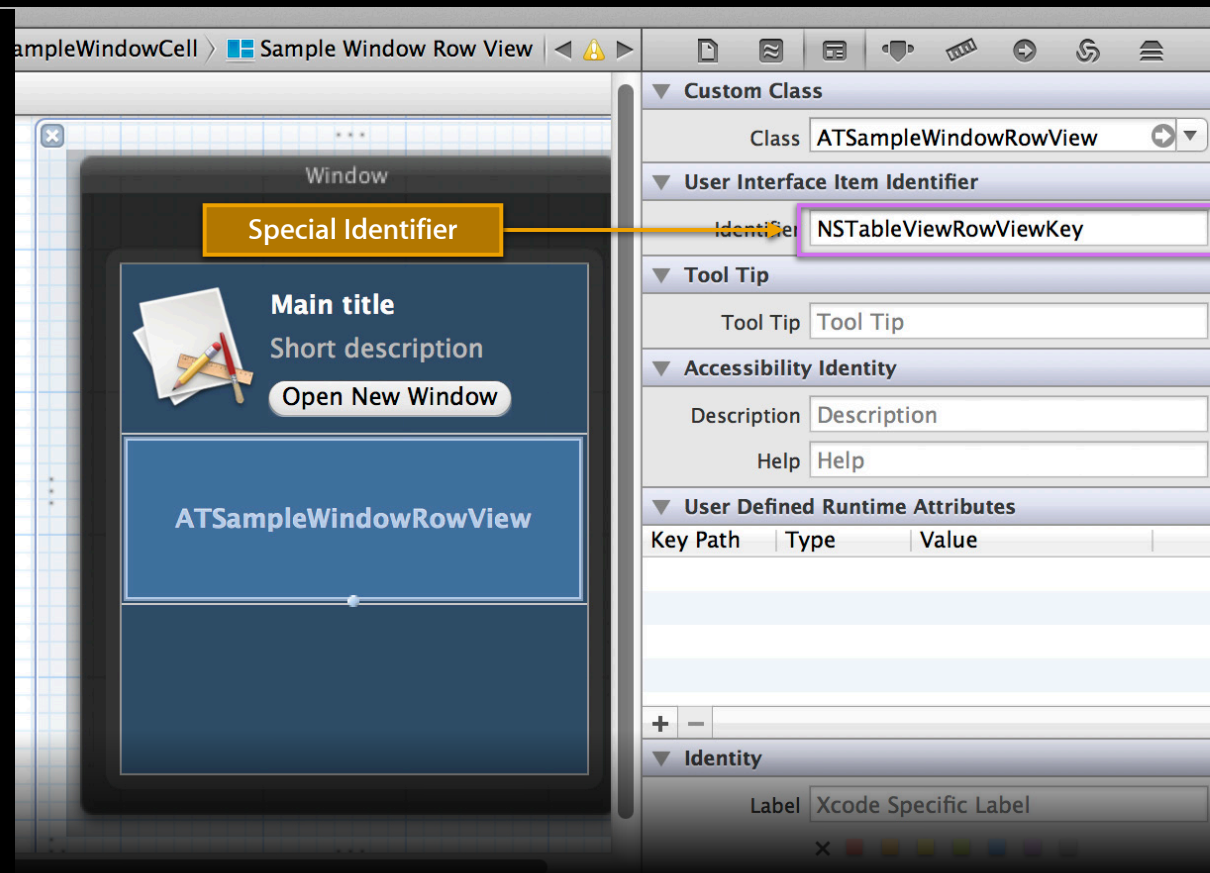
Can also be done at design time

```
- (NSTableView *)tableView:(NSTableView *)tableView  
    tableViewForRow:(NSInteger)row {  
    MyRowView *result =  
        [tableView makeViewWithIdentifier:@"RowView" owner:self];  
    // Setup custom properties on the row view  
    return [result autorelease];  
}
```

All this is optional!

Automatic Row Views

NSTableViewRowViewKey



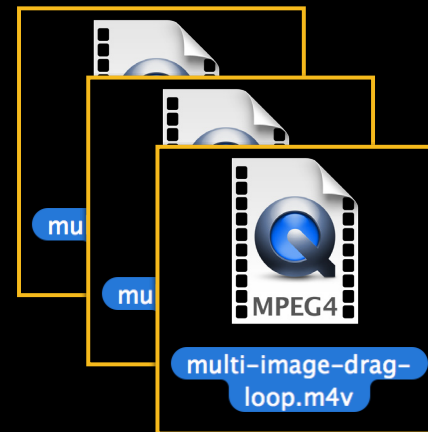
Demo

Drag and drop in TableViewPlayground

Multi-Image Dragging

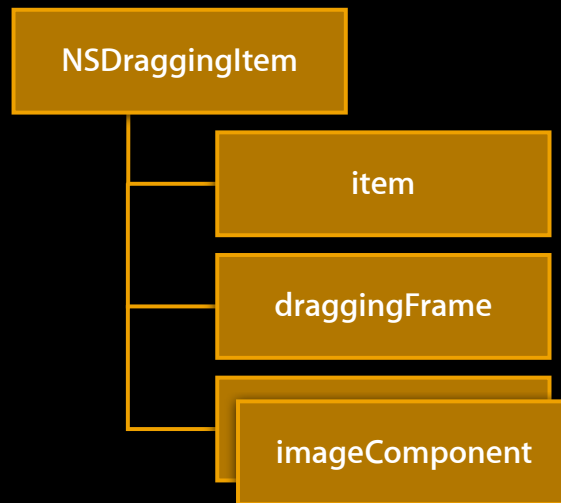
New

- Previously one drag image
- Now multiple NSDraggingItems



Multi-Image Dragging

NSDraggingItem components



Multi-Image Dragging

```
@interface NSDraggingItem : NSObject { ... }  
  
- (id)initWithPasteboardWriter:(id <NSPasteboardWriting>)pasteboardWriter;  
  
@property(copy) NSArray *(^imageComponentsProvider)(void);  
  
@end
```

Array of NSDraggingImageComponent

Placing Items on the Pasteboard

Using NSPasteboardWriter

```
- (id <NSPasteboardWriting>)tableView:(NSTableView *)tableView  
    pasteboardWriterForRow:(NSInteger)row {  
    // Support for us being a dragging source  
    return (ATDesktopEntity *)[_tableContents objectAtIndex:row];  
}
```

Placing Items on the Pasteboard

Using NSPasteboardWriter

```
- (id <NSPasteboardWriting>)tableView:(NSTableView *)tableView  
    pasteboardWriterForRow:(NSInteger)row {  
    // Support for us being a dragging source  
    return (ATDesktopEntity *)[_tableContents objectAtIndex:row];  
}
```

NSTableView creates NSDraggingItem using
the NSPasteboardWriter

Placing Items on the Pasteboard

Using NSPasteboardWriter

```
- (id <NSPasteboardWriting>)tableView:(NSTableView *)tableView  
    pasteboardWriterForRow:(NSInteger)row {  
    // Support for us being a dragging source  
    return (ATDesktopEntity *)[_tableContents objectAtIndex:row];  
}
```

- Return nil to prevent dragging

Placing Items on the Pasteboard

Implementing NSPasteboardWriter

```
@interface ATDesktopEntity :  
    NSObject<NSPasteboardWriting, NSPasteboardReading> {  
  
}  
  
@property (retain) NSURL *fileURL;  
  
@end
```

Placing Items on the Pasteboard

Implementing NSPasteboardWriter

```
@interface ATDesktopEntity :  
    NSObject<NSPasteboardWriting, NSPasteboardReading> {  
  
}  
  
@property (retain) NSURL *fileURL;  
  
@end
```

Placing Items on the Pasteboard

Implementing NSPasteboardWriter

```
- (NSArray *)writableTypesForPasteboard:(NSPasteboard *)pasteboard {  
    return [self.fileURL writableTypesForPasteboard:pasteboard];  
}  
  
- (id)pasteboardPropertyListForType:(NSString *)type {  
    return [self.fileURL pasteboardPropertyListForType:type];  
}
```

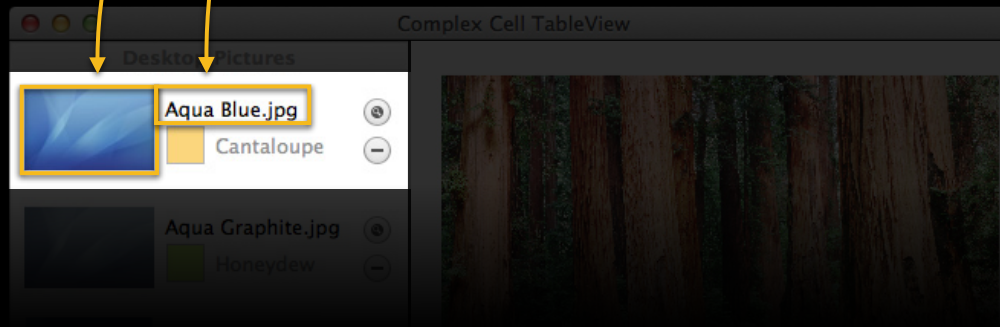
Placing Items on the Pasteboard

Implementing NSPasteboardWriter

```
- (NSArray *)writableTypesForPasteboard:(NSPasteboard *)pasteboard {  
    return [self.fileURL writableTypesForPasteboard:pasteboard];  
}  
  
- (id)pasteboardPropertyListForType:(NSString *)type {  
    return [self.fileURL pasteboardPropertyListForType:type];  
}
```

Providing Drag Images

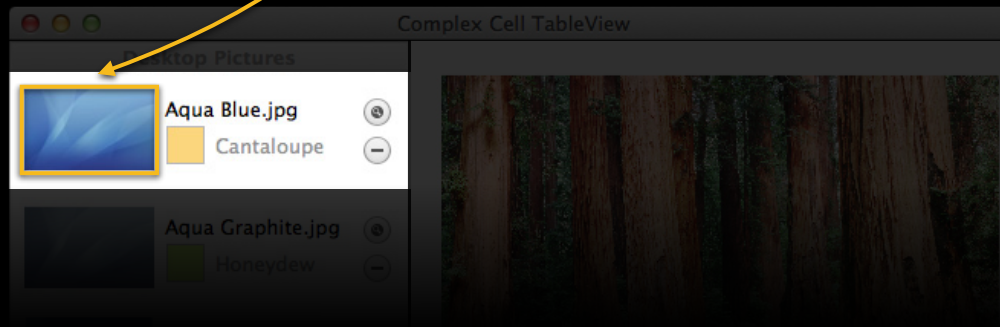
```
@interface NSDraggingItem : NSObject { ... }  
@property(copy) NSArray *(^imageComponentsProvider)(void);  
@end
```



Providing Drag Images

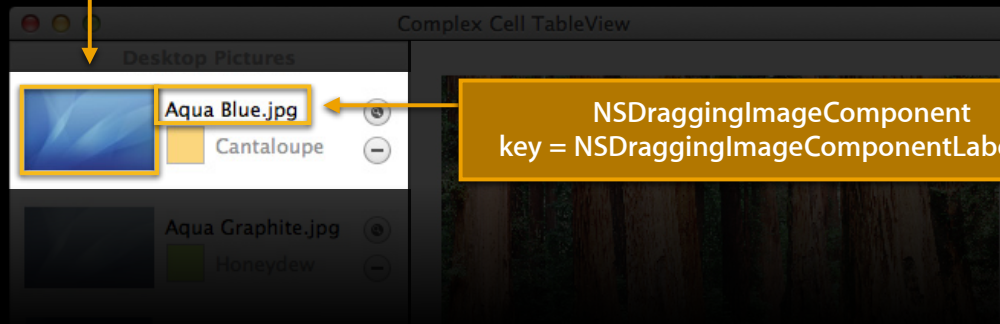
```
@interface NSDraggingItem : NSObject { ... }  
@property(copy) NSArray *(^imageComponentsProvider)(void);  
@end
```

```
@interface NSDraggingImageComponent : NSObject { ... }  
- (id)initWithKey:(NSString *)key;  
@property(retain) id contents; // Typically an NSImage  
...  
@end
```



Providing Drag Images

NSDraggingImageComponent
key = NSDraggingImageComponentIconKey

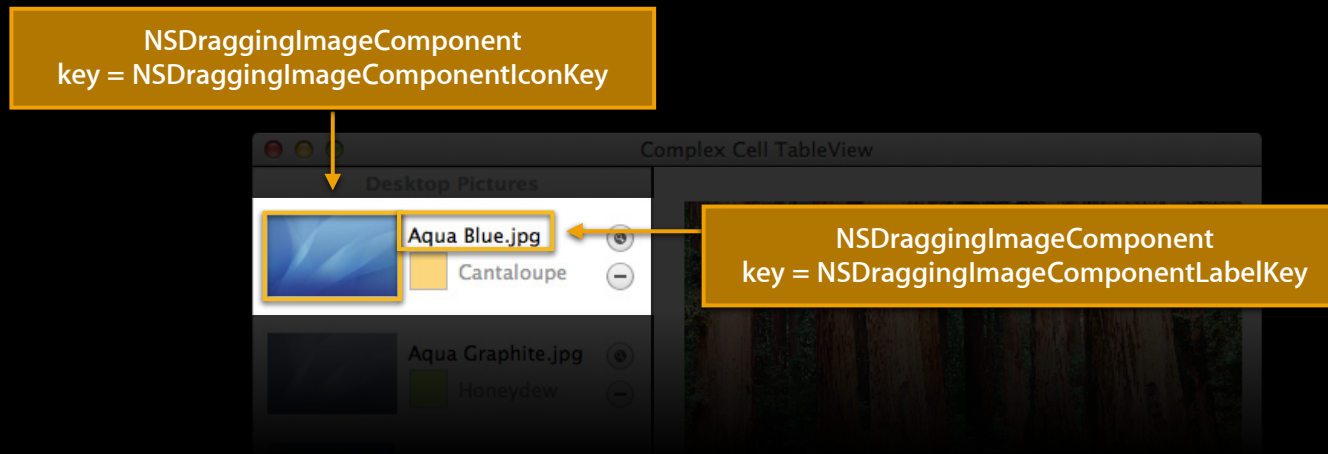


NSDraggingImageComponent
key = NSDraggingImageComponentLabelKey

Providing Drag Images

- Automatically provided by NSTableView:

```
@property(retain, readonly) NSArray *draggingImageComponents;
```

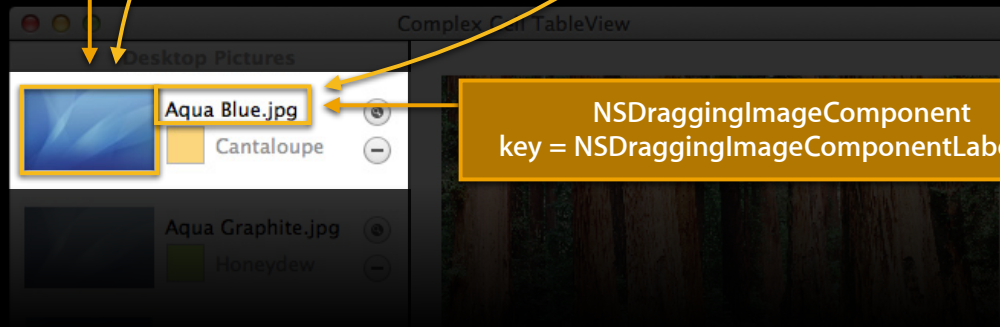


Providing Drag Images

- Automatically provided by NSTableView:

```
@interface NSTableView : NSView { ... }  
@property(assign) IBOutlet NSTextField *textField;  
@property(assign) IBOutlet UIImageView *imageView;
```

NSDraggingImageComponent
key = NSDraggingImageComponentIconKey

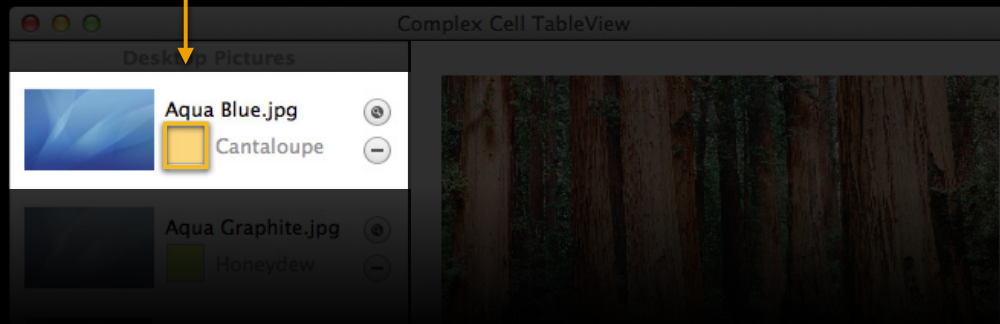


NSDraggingImageComponent
key = NSDraggingImageComponentLabelKey

Providing Drag Images

- How do you add other items?

NSDraggingImageComponent
key = @"Color"



Providing Drag Images

Subclass NSTableView

```
- (NSArray *)draggingImageComponents {
    NSMutableArray *result =
        [[[super draggingImageComponents] mutableCopy] autorelease];

    NSDraggingImageComponent *component =
        [NSDraggingImageComponent draggingImageComponentWithKey:@"Color"];
    component.contents = // NSImage created from the custom color view
    component.frame = [self convertRect:colorView.bounds fromView:colorView];

    [result addObject:component];
    return result;
}
```

Providing Drag Images

Subclass NSTableView

```
- (NSArray *)draggingImageComponents {
    NSMutableArray *result =
        [[[super draggingImageComponents] mutableCopy] autorelease];

    NSDraggingImageComponent *component =
        [NSDraggingImageComponent draggingImageComponentWithKey:@"Color"];
    component.contents = // NSImage created from the custom color view
    component.frame = [self convertRect:colorView.bounds fromView:colorView];

    [result addObject:component];
    return result;
}
```

Providing Drag Images

Subclass NSTableView

```
- (NSArray *)draggingImageComponents {
    NSMutableArray *result =
        [[[super draggingImageComponents] mutableCopy] autorelease];

    NSDraggingImageComponent *component =
        [NSDraggingImageComponent draggingImageComponentWithKey:@"Color"];
    component.contents = // NSImage created from the custom color view
    component.frame = [self convertRect:colorView.bounds fromView:colorView];

    [result addObject:component];
    return result;
}
```

Providing Drag Images

Subclass NSTableView

```
- (NSArray *)draggingImageComponents {
    NSMutableArray *result =
        [[[super draggingImageComponents] mutableCopy] autorelease];

    NSDraggingImageComponent *component =
        [NSDraggingImageComponent draggingImageComponentWithKey:@"Color"];
    component.contents = // NSImage created from the custom color view
    component.frame = [self convertRect:colorView.bounds fromView:colorView];

    [result addObject:component];
    return result;
}
```

Changing Drag Images





















The screenshot shows a macOS window titled "Complex OutlineView" with a table of desktop images and a color palette. The table has columns for "Desktop Image", "Date Modified", and "Reveal" buttons. The "Eagle & Waterfall.jpg" row is selected. A color palette is visible on the right, with a "Solid Kelp.png" image being dragged over the "Fern" color swatch.

Desktop Image	Date Modified
Andromeda Galaxy.jpg	May 17, 2011
Beach.jpg	May 17, 2011
Bristle Grass.jpg	May 17, 2011
Ducks on a Misty Pond.jpg	May 17, 2011
Eagle & Waterfall.jpg	May 17, 2011
Elephant.jpg	May 17, 2011
Flamingos.jpg	May 17, 2011
Floating Leaves.jpg	May 17, 2011
Forest in Mist.jpg	May 17, 2011
Grass Blades.jpg	May 17, 2011
Isles.jpg	May 17, 2011
Lake.jpg	May 17, 2011
Lion.jpg	May 17, 2011
Moon.jpg	May 17, 2011
Mt. Fuji.jpg	May 17, 2011
Pink Forest.jpg	May 17, 2011
Pink Lotus Flower.jpg	May 17, 2011
Poppies.jpg	May 17, 2011
Red Bells.jpg	May 17, 2011
Solid Colors	
Solid Aqua Blue.png	May 9, 2011

Color Palette:

- Solid Kelp.png
- Magnesium
- Mocha
- Fern**
- Moss
- Ocean
- Eggplant
- Maroon
- Steel
- Aluminum
- Cayenne
- Asparagus
- Clover
- Teal
- Midnight
- Plum
- Tin
- Nickel
- Cantaloupe

Changing Drag Images

Complex OutlineView			
Desktop Image		Date Modified	Color
	Andromeda Galaxy.jpg	X Reveal May 17, 2011	Magenta
	Beach.jpg	X Reveal May 17, 2011	Iron
	Bristle Grass.jpg	X Reveal May 17, 2011	Magnesium
	Ducks on a Misty Pond.jpg	X Reveal May 17, 2011	Mocha
	Eagle & Waterfall.jpg	X Reveal May 17, 2011	Fern
	Elephant.jpg	X Reveal May 17, 2011	Moss
	Flamingos.jpg	X Reveal May 17, 2011	Ocean
	Floating Leaves.jpg	X Reveal May 17, 2011	Eggplant
	Forest in Mist.jpg	X Reveal May 17, 2011	Maroon
	Grass Blades.jpg	X Reveal May 17, 2011	Steel
	Isles.jpg	X Reveal May 17, 2011	Aluminum
	Lake.jpg	X Reveal May 17, 2011	Cayenne
	Lion.jpg	X Reveal May 17, 2011	Asparagus
	Moon.jpg	X Reveal May 17, 2011	Clover
	Mt. Fuji.jpg	X Reveal May 17, 2011	Teal
	Pink Forest.jpg	X Reveal May 17, 2011	Midnight
	Pink Lotus Flower.jpg	X Reveal May 17, 2011	Plum
	Poppies.jpg	X Reveal May 17, 2011	Tin
	Red Bells.jpg	X Reveal May 17, 2011	Nickel
Solid Colors			
	Solid Aqua Blue.png	X Reveal May 9, 2011	Cantaloupe

Changing Drag Images



NSTableView/NSOutlineView delegate method

```
- (void)outlineView:(NSOutlineView *)outlineView  
    updateDraggingItemsForDrag:(id <NSDraggingInfo>)draggingInfo {
```

Changing Drag Images

Create a basic view to “draw” with

```
- (void)outlineView:(NSOutlineView *)outlineView
    updateDraggingItemsForDrag:(id <NSDraggingInfo>)draggingInfo {

    ATableViewCell *tableView =
        [outlineView makeViewWithIdentifier:@"MainCell" owner:self];
    CGRect cellFrame = CGRectMake(0, 0,
        [tableView width], [tableView height]);
```

Changing Drag Images

Enumerate the source dragging items

...

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
[draggingInfo enumerateDraggingItemsWithOptions:0
 forView:_outlineView classes:classes searchOptions:nil
 usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    draggingItem.draggingFrame = cellForme;
    draggingItem.imageComponentsProvider = ^(void) {
        tableView.objectValue = draggingItem.item;
        tableView.frame = cellForme;
        return [tableView draggingImageComponents];
    };
}
```

Changing Drag Images

Enumerate the source dragging items

...

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
[draggingInfo enumerateDraggingItemsWithOptions:0
    forView:_outlineView classes:classes searchOptions:nil
    usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    draggingItem.draggingFrame = cellForme;
    draggingItem.imageComponentsProvider = ^(void) {
        tableView.objectValue = draggingItem.item;
        tableView.frame = cellForme;
        return [tableView draggingImageComponents];
    };
}
```

Changing Drag Images

Enumerate the source dragging items

...

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
[draggingInfo enumerateDraggingItemsWithOptions:0
 forView:_outlineView classes:classes searchOptions:nil
 usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    draggingItem.draggingFrame = cellForFrame;
    draggingItem.imageComponentsProvider = ^(void) {
        tableView.objectValue = draggingItem.item;
        tableView.frame = cellForFrame;
        return [tableView draggingImageComponents];
    };
}
```

Changing Drag Images

Enumerate the source dragging items

...

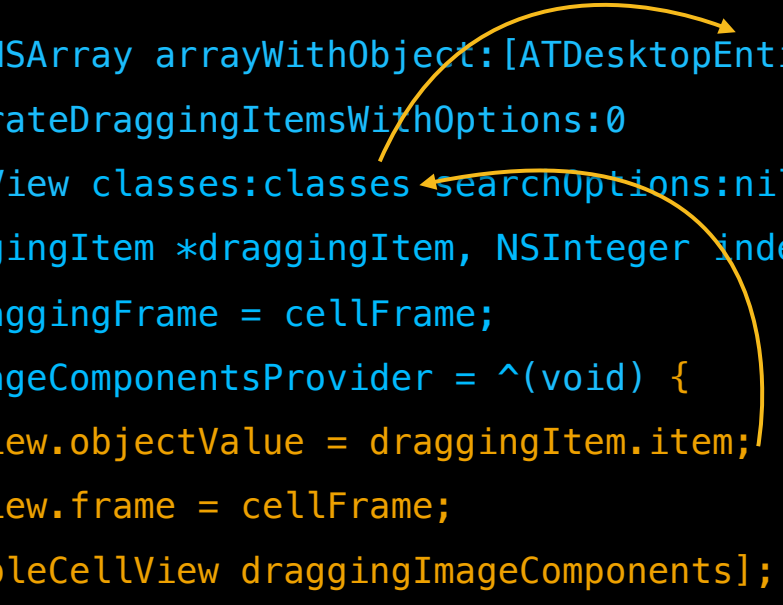
```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
[draggingInfo enumerateDraggingItemsWithOptions:0
 forView:_outlineView classes:classes searchOptions:nil
 usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    draggingItem.draggingFrame = cellForme;
    draggingItem.imageComponentsProvider = ^(void) {
        tableView.objectValue = draggingItem.item;
        tableView.frame = cellForme;
        return [tableView draggingImageComponents];
    };
}
```

Changing Drag Images

Enumerate the source dragging items

...

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
[draggingInfo enumerateDraggingItemsWithOptions:0
 forView:_outlineView classes:classes searchOptions:nil
 usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    draggingItem.draggingFrame = cellForme;
    draggingItem.imageComponentsProvider = ^(void) {
        tableView.objectValue = draggingItem.item;
        tableView.frame = cellForme;
        return [tableView draggingImageComponents];
    };
};
}
```

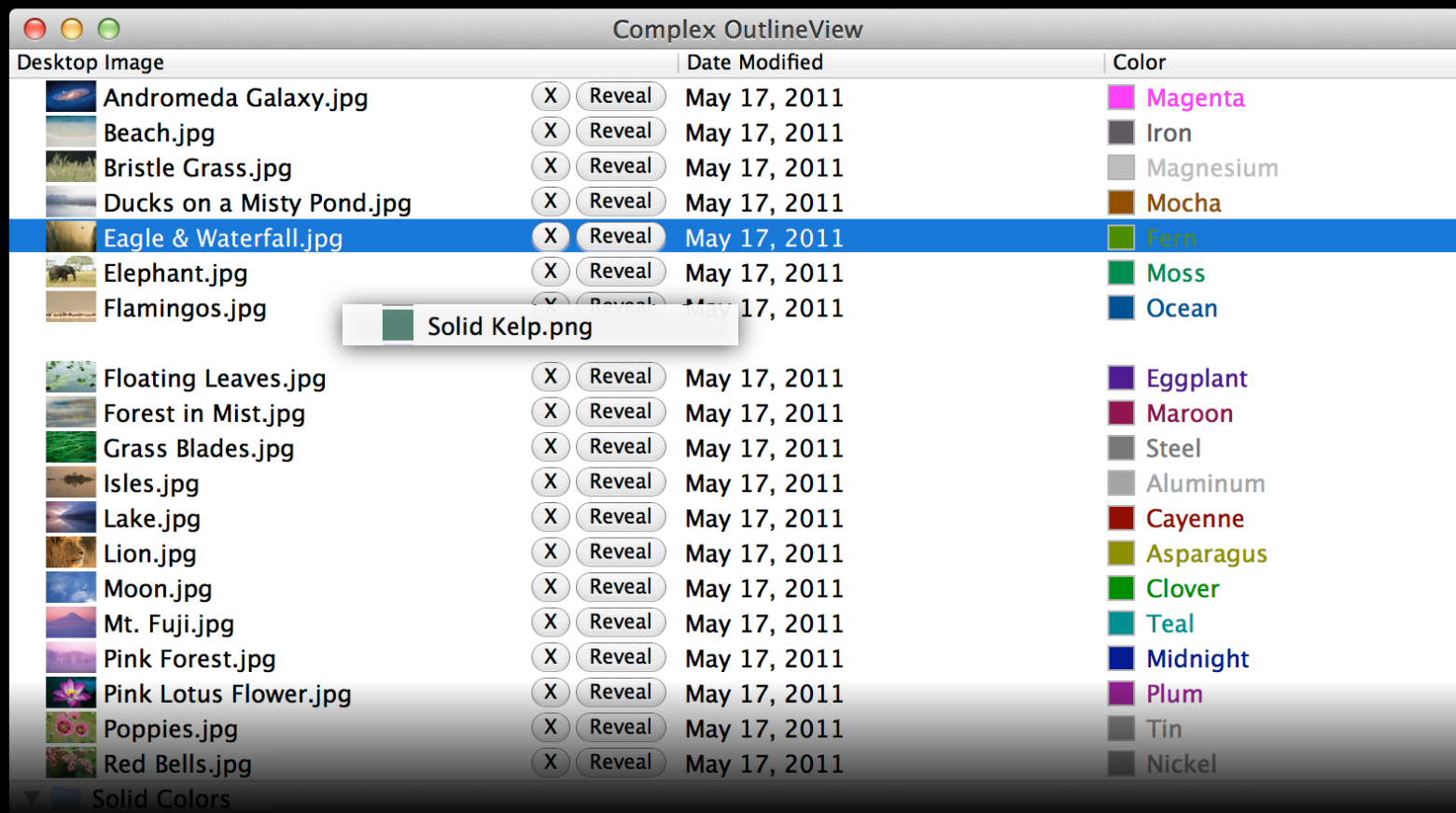


Accepting a Drop Along with an animation

The screenshot shows a macOS window titled "Complex OutlineView" with a table of files. The table has three columns: "Desktop Image", "Date Modified", and "Color". A file named "Solid Kelp.png" is being dragged over the table, indicated by a tooltip above it. The file "Eagle & Waterfall.jpg" is currently selected.

Desktop Image	Date Modified	Color
Andromeda Galaxy.jpg	X Reveal May 17, 2011	Magenta
Beach.jpg	X Reveal May 17, 2011	Iron
Bristle Grass.jpg	X Reveal May 17, 2011	Magnesium
Ducks on a Misty Pond.jpg	X Reveal May 17, 2011	Mocha
Eagle & Waterfall.jpg	X Reveal May 17, 2011	Fern
Elephant.jpg	X Reveal May 17, 2011	Moss
Flamingos.jpg	X Reveal May 17, 2011	Ocean
Floating Leaves.jpg	X Reveal May 17, 2011	Eggplant
Forest in Mist.jpg	X Reveal May 17, 2011	Maroon
Grass Blades.jpg	X Reveal May 17, 2011	Steel
Isles.jpg	X Reveal May 17, 2011	Aluminum
Lake.jpg	X Reveal May 17, 2011	Cayenne
Lion.jpg	X Reveal May 17, 2011	Asparagus
Moon.jpg	X Reveal May 17, 2011	Clover
Mt. Fuji.jpg	X Reveal May 17, 2011	Teal
Pink Forest.jpg	X Reveal May 17, 2011	Midnight
Pink Lotus Flower.jpg	X Reveal May 17, 2011	Plum
Poppies.jpg	X Reveal May 17, 2011	Tin
Red Bells.jpg	X Reveal May 17, 2011	Nickel
▼ Solid Colors		
Solid Aqua Blue.png	X Reveal May 9, 2011	Cantaloupe

Accepting a Drop Along with an animation



First: Drop Validation

Tell the drag system you want to animate

```
- (NSDragOperation)outlineView:(NSOutlineView *)outlineView
    validateDrop:(id <NSDraggingInfo>)info
    proposedItem:(id)item
    proposedChildIndex:(NSInteger)index {

    // Typical drop validation based on the pasteboard
    info.animatesToDestination = YES;
    return NSDragOperationCopy; // or Move, etc.

}
```

First: Drop Validation

Tell the drag system you want to animate

```
- (NSDragOperation)outlineView:(NSOutlineView *)outlineView
    validateDrop:(id <NSDraggingInfo>)info
    proposedItem:(id)item
    proposedChildIndex:(NSInteger)index {

    // Typical drop validation based on the pasteboard
    info.animatesToDestination = YES;
    return NSDragOperationCopy; // or Move, etc.

}
```

First: Drop Validation

Tell the drag system you want to animate

```
- (NSDragOperation)outlineView:(NSOutlineView *)outlineView
    validateDrop:(id <NSDraggingInfo>)info
    proposedItem:(id)item
    proposedChildIndex:(NSInteger)index {

    // Typical drop validation based on the pasteboard
    info.animatesToDestination = YES;
    return NSDragOperationCopy; // or Move, etc.

}
```



Next: Accepting a Drop

Use the existing delegate method

```
- (BOOL)outlineView:(NSOutlineView *)outlineView
    acceptDrop:(id <NSDraggingInfo>)info
        item:(ATDesktopEntity *)item
    childIndex:(NSInteger)childIndex {

    ...

}
```

Accepting a Drop

Enumerate the draggingItems

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
__block NSInteger insertionIndex = childIndex;
[info enumerateDraggingItemsWithOptions:0
                    forView:_outlineView
                    classes:classes
                    searchOptions:nil
                    usingBlock:
    ^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ...
}
```

Accepting a Drop

Enumerate the draggingItems

```
NSArray *classes = [NSArray arrayWithObject:[ATDesktopEntity class]];
__block NSInteger insertionIndex = childIndex;
[info enumerateDraggingItemsWithOptions:0
        forView:_outlineView
        classes:classes
        searchOptions:nil
        usingBlock:
    ^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ...
}
```

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                                row:row];

    insertionIndex++;
}];
```


Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndexesInRange:NSMakeRange(index, 1)];
    [_outlineView insertItemsAtIndexes:indexes
                           inParent:target
                           withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                             row:row];

    insertionIndex++;
}];
```

From the classes array.
Created with NSPasteboardReader

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [target.children insertObjects:draggingItem.itemsAtIndexes:indexes
                               inParent:target
                               withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                                row:row];

    insertionIndex++;
}];
```

This is the model object

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSInteger *indexes = [NSMutableArray arrayWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                                row:row];

    insertionIndex++;
}];
```

Insert it into the view object

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView numberOfRows];
    draggingItem.draggingFrame = CGRectMake(0, row * 20, 100, 20);
    [_outlineView insertItemAtIndex:insertionIndex Column:colIndex
                          row:row];

    insertionIndex++;
}];
```

Leaves a gap during the animation
and pops the row in at the end

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationNone];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                              row:row];

    insertionIndex++;
}];
```

At this point, table is in its "final" state

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                                row:row];

    insertionIndex++;
}];
```

Accepting a Drop

Enumerate the draggingItems

```
usingBlock:^(NSDraggingItem *draggingItem, NSInteger index, BOOL *stop) {
    ATDesktopEntity *entity = (ATDesktopEntity *)draggingItem.item;
    [target.children insertObject:entity atIndex:insertionIndex];
    NSMutableIndexSet *indexes = [NSMutableIndexSet indexSetWithIndex:insertionIndex];
    [_outlineView insertItemsAtIndexes:indexes
                          inParent:target
                          withAnimation:NSTableViewAnimationEffectGap];

    NSInteger row = [_outlineView rowForItem:entity];
    draggingItem.draggingFrame = [_outlineView frameOfCellAtColumn:colIndex
                                                              row:row];

    insertionIndex++;
}];
```

Tell the drag image where to go

Animating

Three Basic Methods

Similar to NSMutableArray

- `(void)insertRowsAtIndexes:(NSIndexSet *)indexes
withAnimation:(NSTableViewAnimationOptions)animationOptions;`
- `(void)removeRowsAtIndexes:(NSIndexSet *)indexes
withAnimation:(NSTableViewAnimationOptions)animationOptions;`
- `(void)moveRowAtIndex:(NSInteger)oldIndex toIndex:(NSInteger)newIndex;`

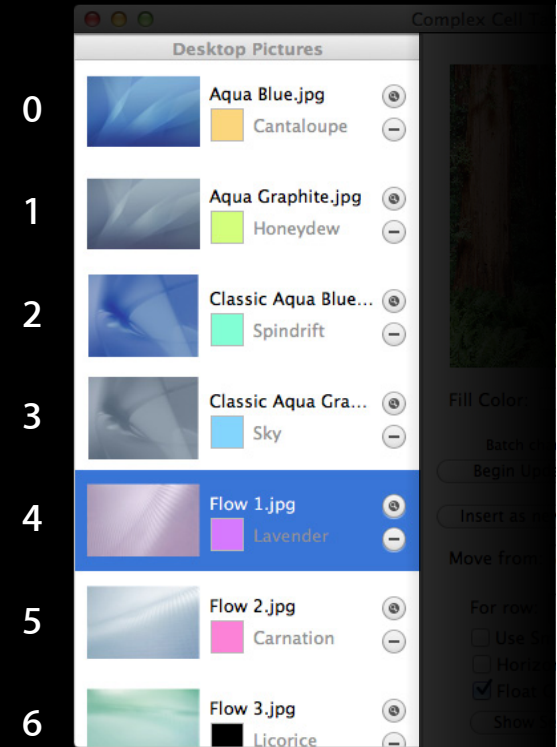
Insertion

Similar to NSMutableArray

- (void)insertRowsAtIndexes: 1, 3 withAnimation: ...

New Row ONE

New Row THREE



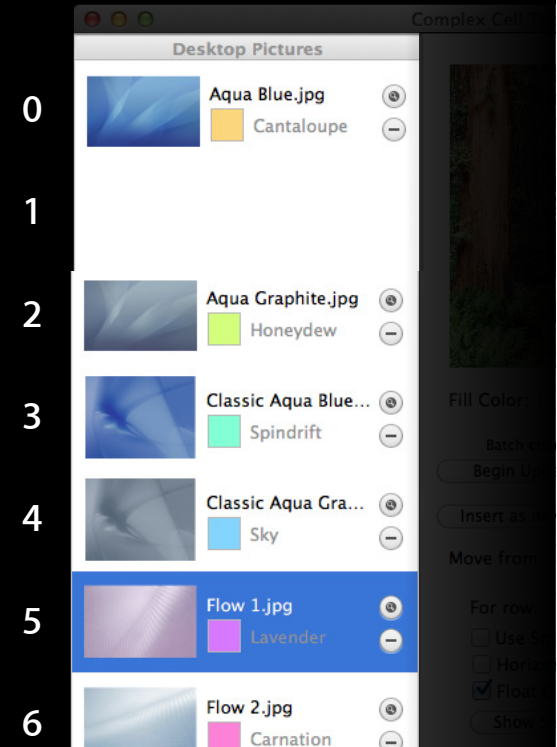
Insertion

Similar to NSMutableArray

- (void)insertRowsAtIndexes: 1, 3 withAnimation: ...

New Row ONE

New Row THREE

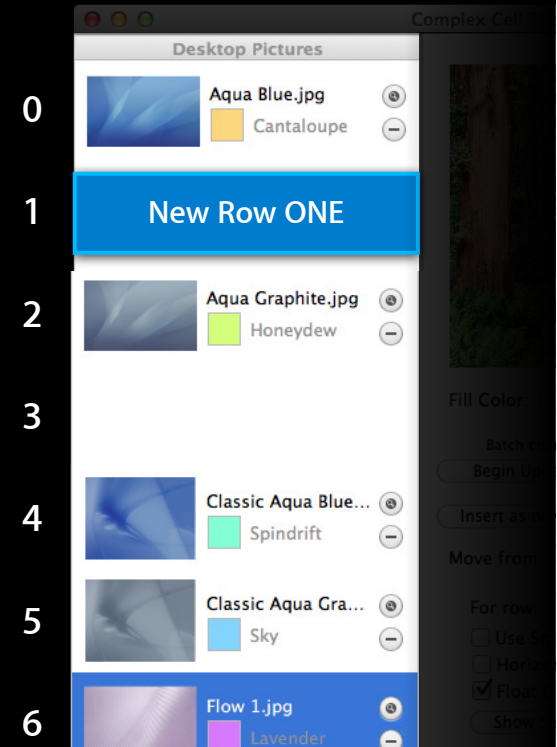


Insertion

Similar to NSMutableArray

- (void)insertRowsAtIndexes: 1, 3 withAnimation: ...

New Row THREE



Batch Updates

Better for performance

```
[table beginUpdates];
```

```
[table insertRowsAtIndexes:... withAnimation:...];
```

```
...
```

```
[table removeRowsAtIndexes:... withAnimation:...];
```

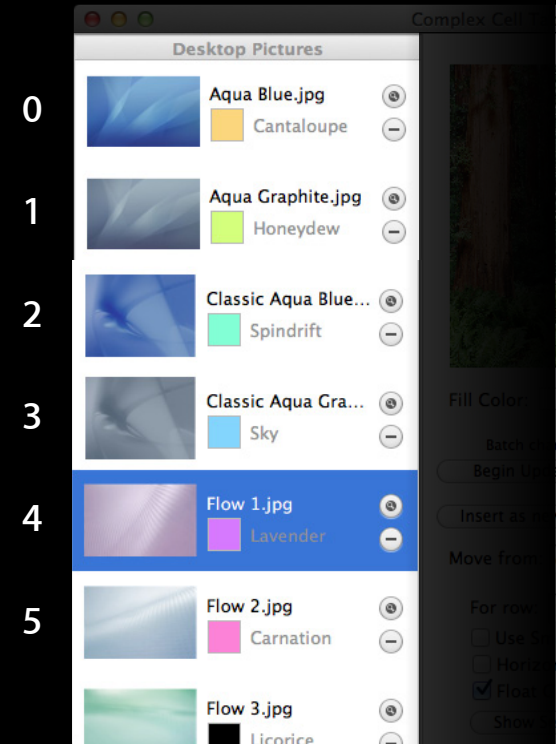
```
[table endUpdates];
```

Batch Updates

Similar to NSMutableArray

```
[table beginUpdates];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table endUpdates];
```

First new row

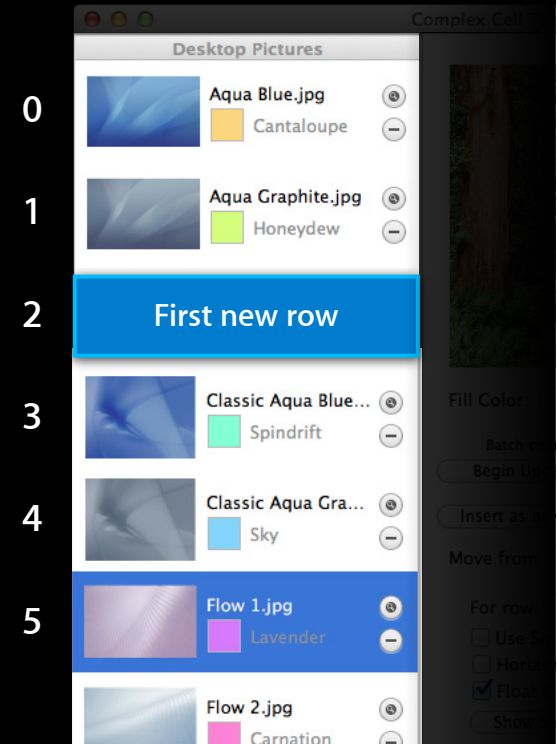


Batch Updates

Similar to NSMutableArray

```
[table beginUpdates];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table endUpdates];
```

Second new row

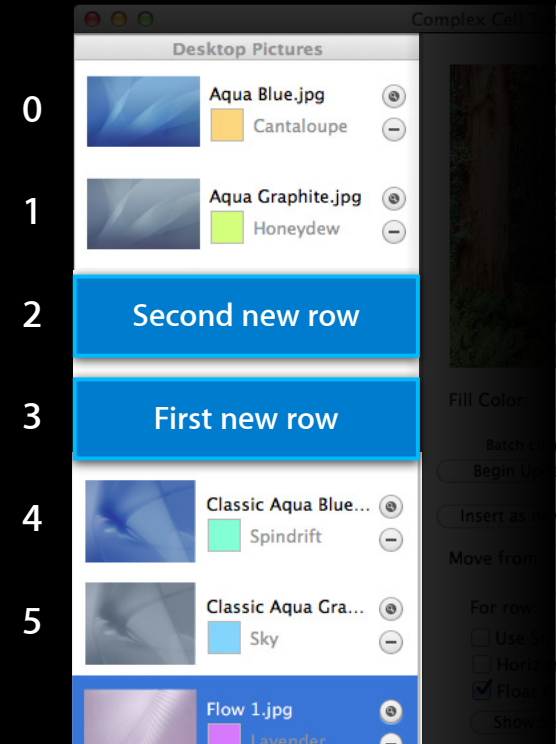


Batch Updates

Similar to NSMutableArray

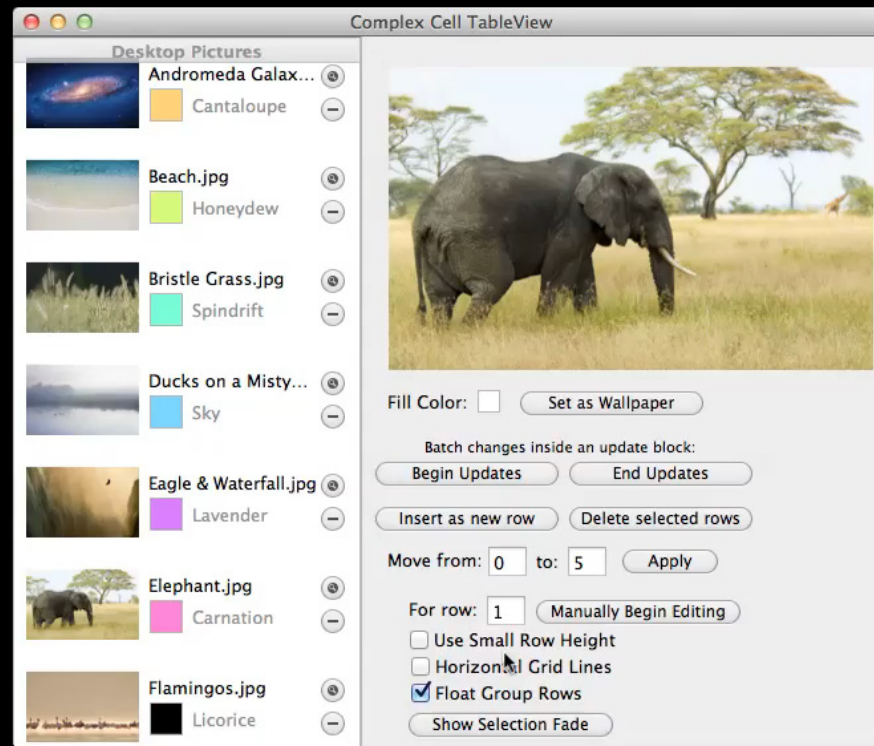
```
[table beginUpdates];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table insertRowsAtIndexes:2 ... ];  
[table endUpdates];
```

Third new row



Row Height Animations

As seen in TableViewPlayground



Row Height Animations

As seen in TableViewPlayground

```
NSIndexPath *indexesToChange = ...;
```

```
[table noteHeightOfRowsWithIndexesChanged:indexesToChange];
```



Always animates for
View-Based TableViews

Syncing Animations

As seen in TableViewPlayground

```
[NSAnimationContext beginGrouping];  
[[NSAnimationContext currentContext] setDuration:1.5];  
  
// Update visible views  
...  
  
[table noteHeightOfRowsWithIndexesChanged:indexes];  
  
[NSAnimationContext endGrouping];
```

Syncing Animations

As seen in TableViewPlayground

```
[NSAnimationContext beginGrouping];  
[[NSAnimationContext currentContext] setDuration:1.5];  
  
// Update visible views  
...  
  
[table noteHeightOfRowsWithIndexesChanged:indexes];  
  
[NSAnimationContext endGrouping];
```

Syncing Animations

As seen in TableViewPlayground

```
[NSAnimationContext beginGrouping];  
[[NSAnimationContext currentContext] setDuration:1.5];  
  
// Update visible views  
...  
  
[table noteHeightOfRowsWithIndexesChanged:indexes];  
  
[NSAnimationContext endGrouping];
```

Disabling Animations

What if you do not want it to animate?

```
[NSAnimationContext beginGrouping];  
[[NSAnimationContext currentContext] setDuration:0];  
  
// Operation that normally animates  
  
[NSAnimationContext endGrouping];
```

Disabling Animations

What if you do not want it to animate?

```
[NSAnimationContext beginGrouping];  
[[NSAnimationContext currentContext] setDuration:0];  
  
// Operation that normally animates  
  
[NSAnimationContext endGrouping];
```

Animations with NSCell

These all work for NSCell-based tables if...

- `(void)insertRowsAtIndexes:(NSIndexSet *)indexes
withAnimation:(NSTableViewAnimationOptions)animationOptions;`
- `(void)removeRowsAtIndexes:(NSIndexSet *)indexes
withAnimation:(NSTableViewAnimationOptions)animationOptions;`
- `(void)moveRowAtIndex:(NSInteger)oldIndex toIndex:(NSInteger)newIndex;`
- `(void)noteHeightOfRowsWithIndexesChanged:(NSIndexSet *)indexes;`

Animations with NSCell

If you use `beginUpdates` and `endUpdates`

```
[table beginUpdates];
```

```
[table insertRowsAtIndexes:... withAnimation:...];
```

```
[table removeRowsAtIndexes:... withAnimation:...];
```

```
[table endUpdates];
```

DragNDropOutlineView Demo Updated

Summary

- Layout
- Construction
- Bindings
- Customizing
- Drag and drop
- Animating

More Information

Bill Dudney

Application Frameworks Evangelist
dudney@apple.com

Documentation

Mac OS X Dev Center
<http://developer.apple.com/devcenter/mac>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

Design Patterns to Simplify Mac Accessibility

Pacific Heights
Thursday 3:15–4:15PM



Labs

Cocoa, Auto Save, File Coordination, and Resume Lab

App Frameworks Lab A
Thursday 2:00–4:15PM

