View-Based NSTableView

Basic to advanced

Session 120

Corbin Dunn
Cocoa Software Engineer

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

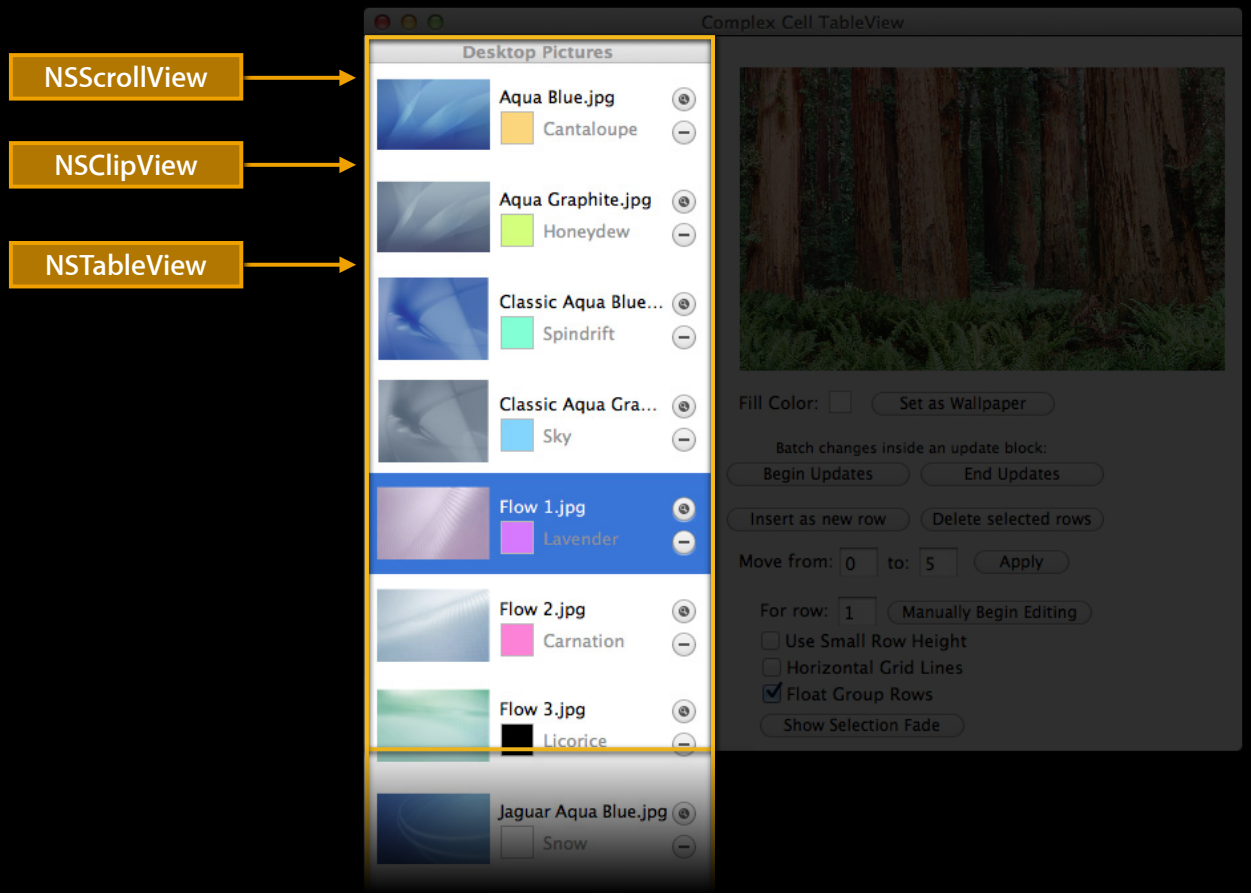
Demo

TableViewPlayground

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

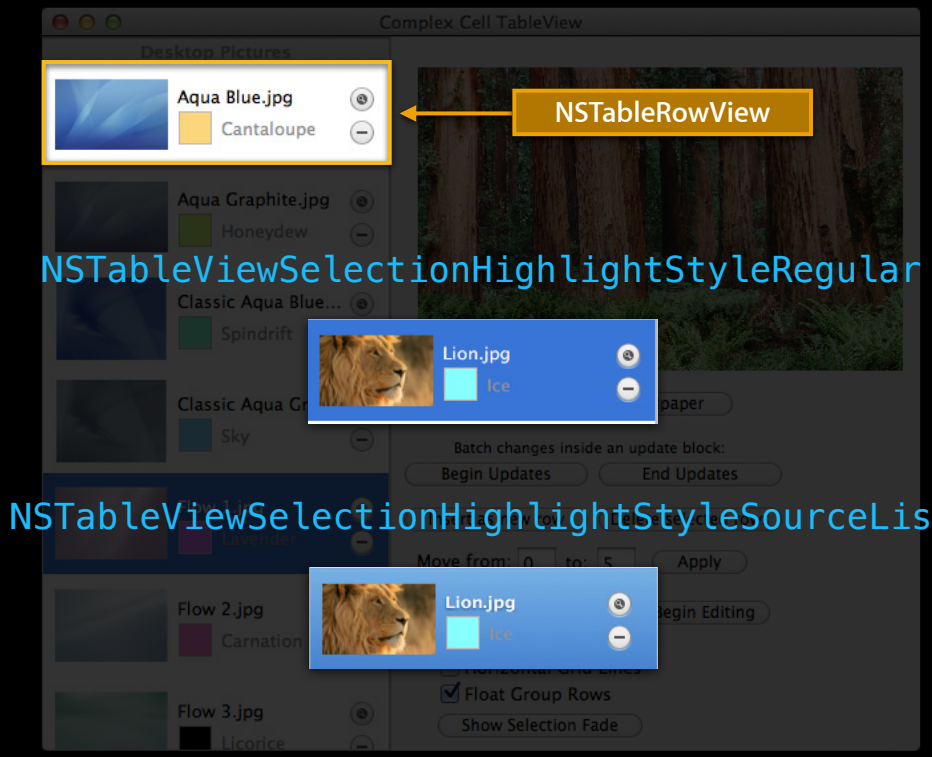
Layout

NSTableView Layout



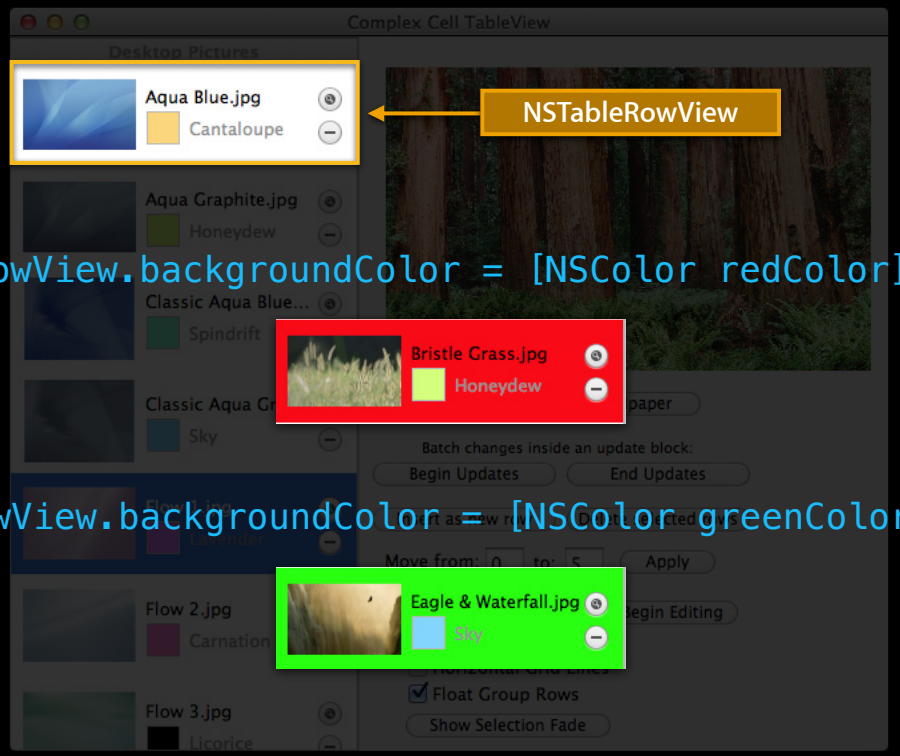
NSTableRowView Layout

- NSTableRowView draws
  - Selection



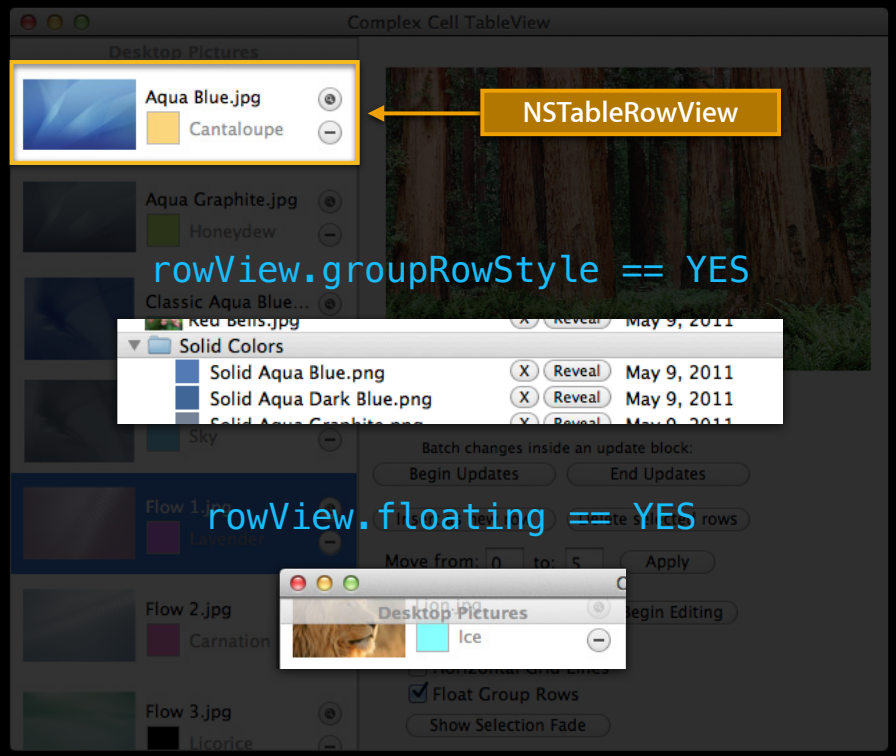
NSTableRowView Layout

- NSTableRowView draws
  - Row background color



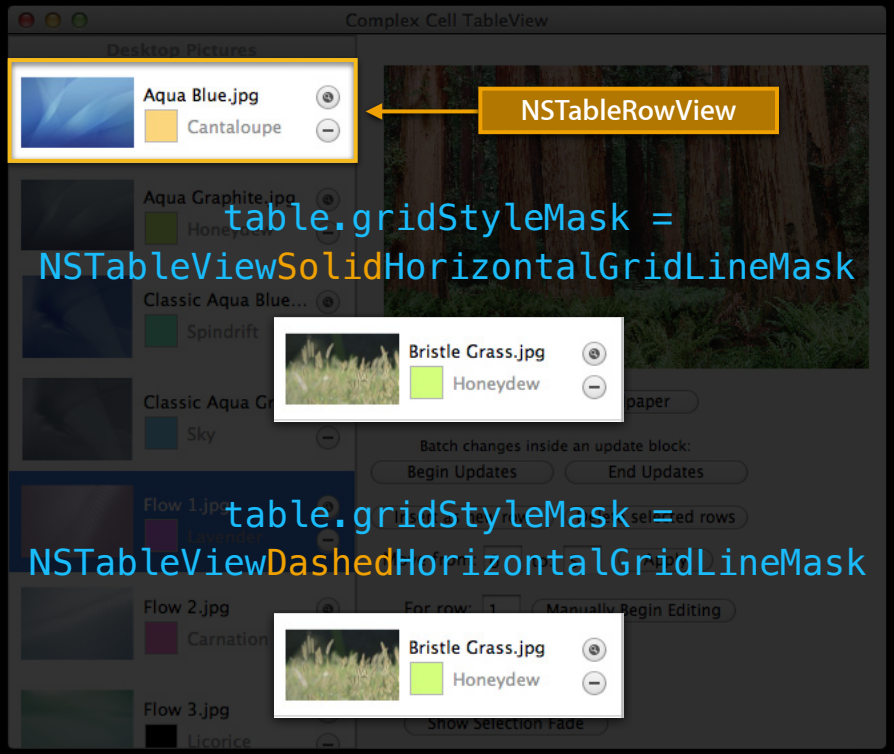
NSTableRowView Layout

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



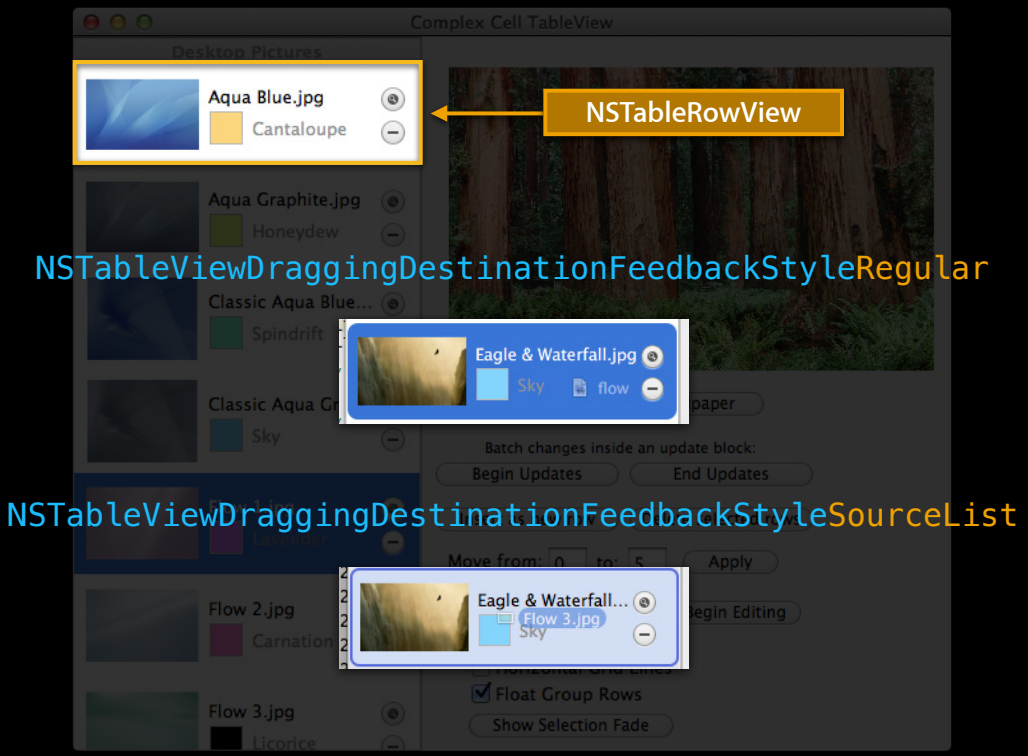
NSTableRowView Layout

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



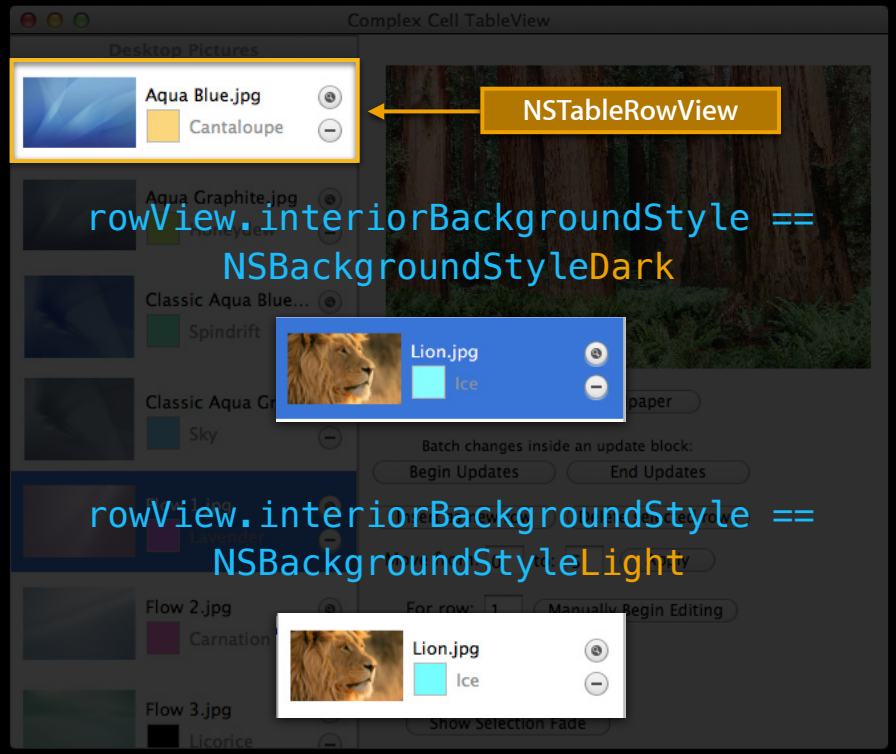
NSTableRowView Layout

- NSTableRowView draws
  - Drag-and-drop feedback



NSTableRowView Layout

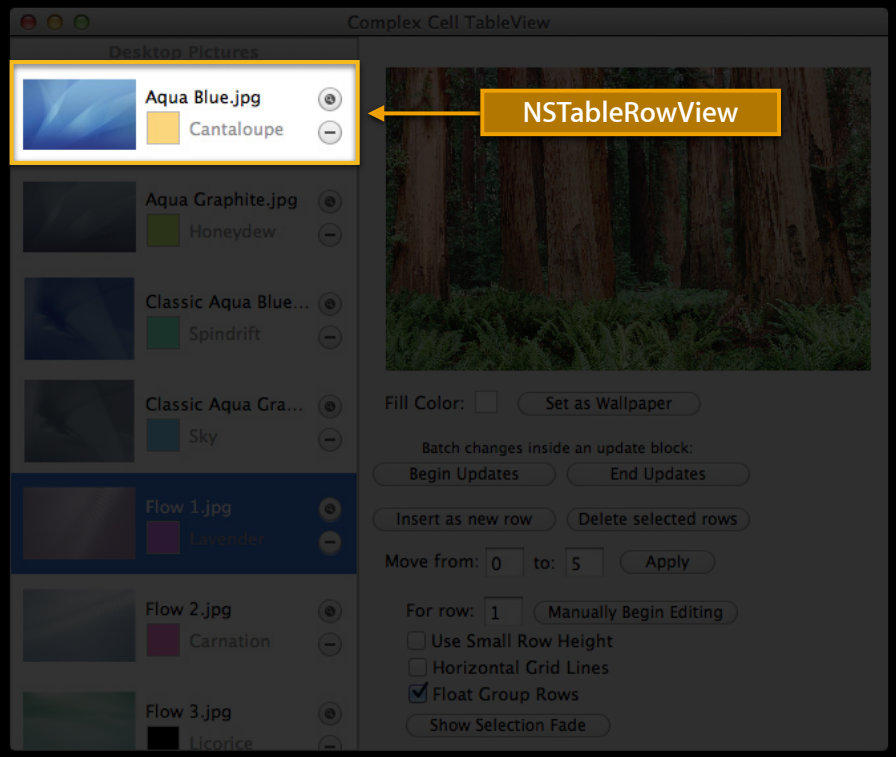
- NSTableRowView computes
  - `-interiorBackgroundStyle` based on set properties



NSTableRowView Layout

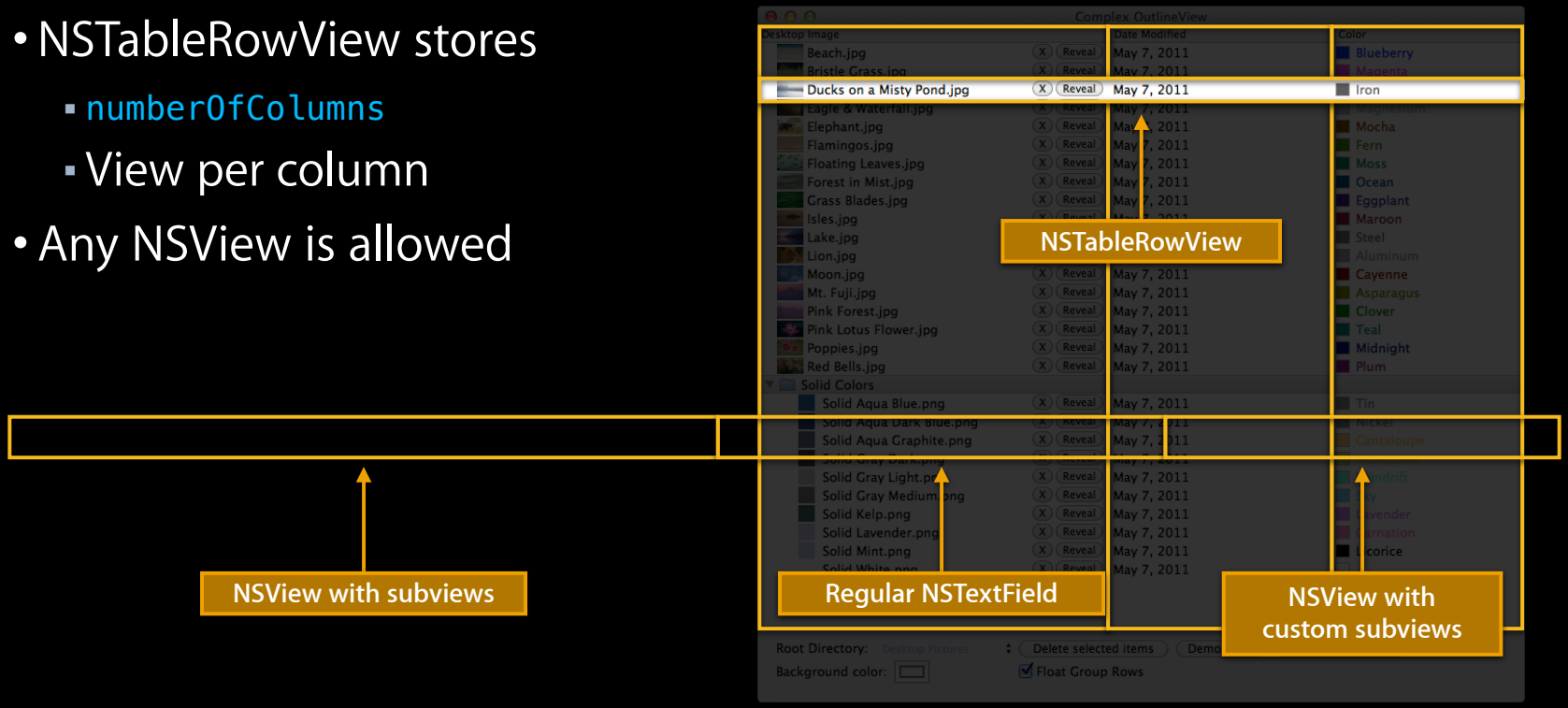
- NSTableRowView does not compute
  - All initial property values are set NSTableView
  - Opportunity to change the values in the delegate

```
-tableView:didAddRowView:forRow:
```



NSTableColumn and NSTableRowView

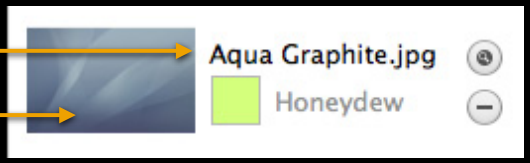
- NSTableRowView stores
  - `numberOfColumns`
  - View per column
- Any NSView is allowed



NSTableCellView

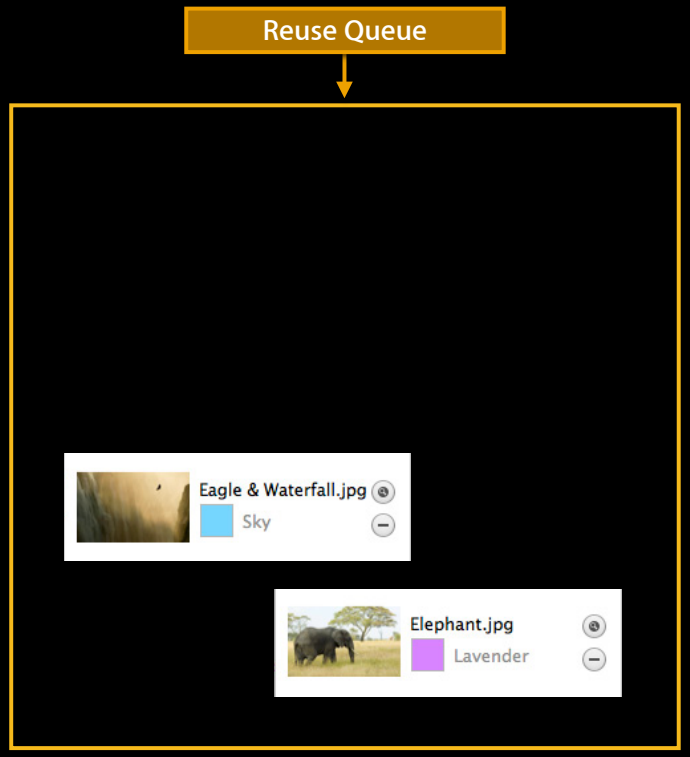
Optional, but allows easy hooks to subviews

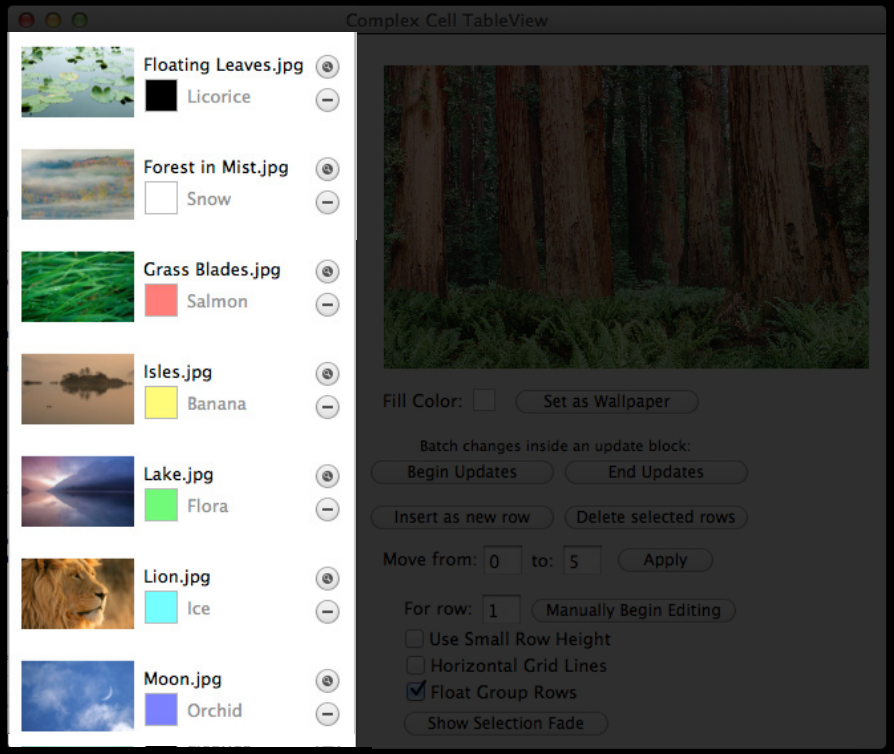
```
@interface NSTableCellView : NSView ...

@property(assign) IBOutlet NSTextField *textField;
@property(assign) IBOutlet NSImageView *imageView;
...
```



Also these are hints for accessibility

How Many Views?





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

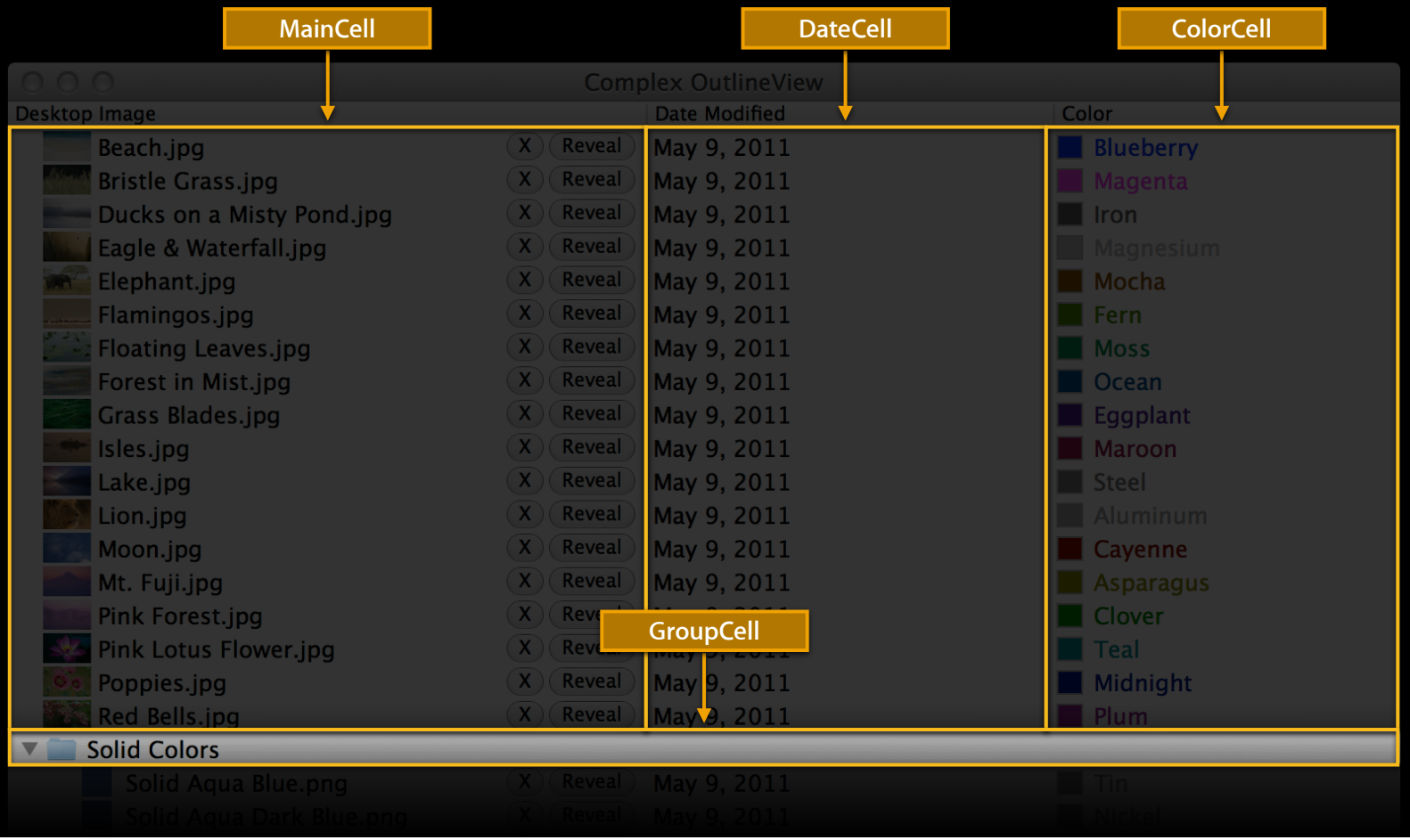
Basic delegate implementation

- 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



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

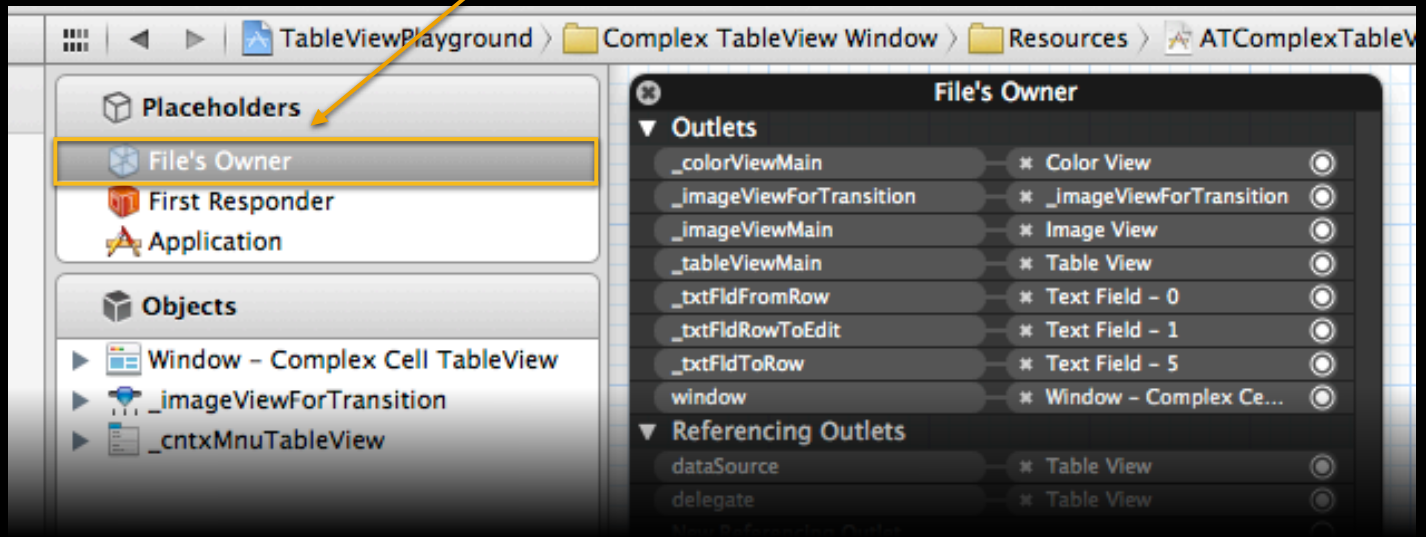
Even more typical implementation with nibs

```
- (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;

}
```

NSNib Loading

Loading an NSView from an NSNib

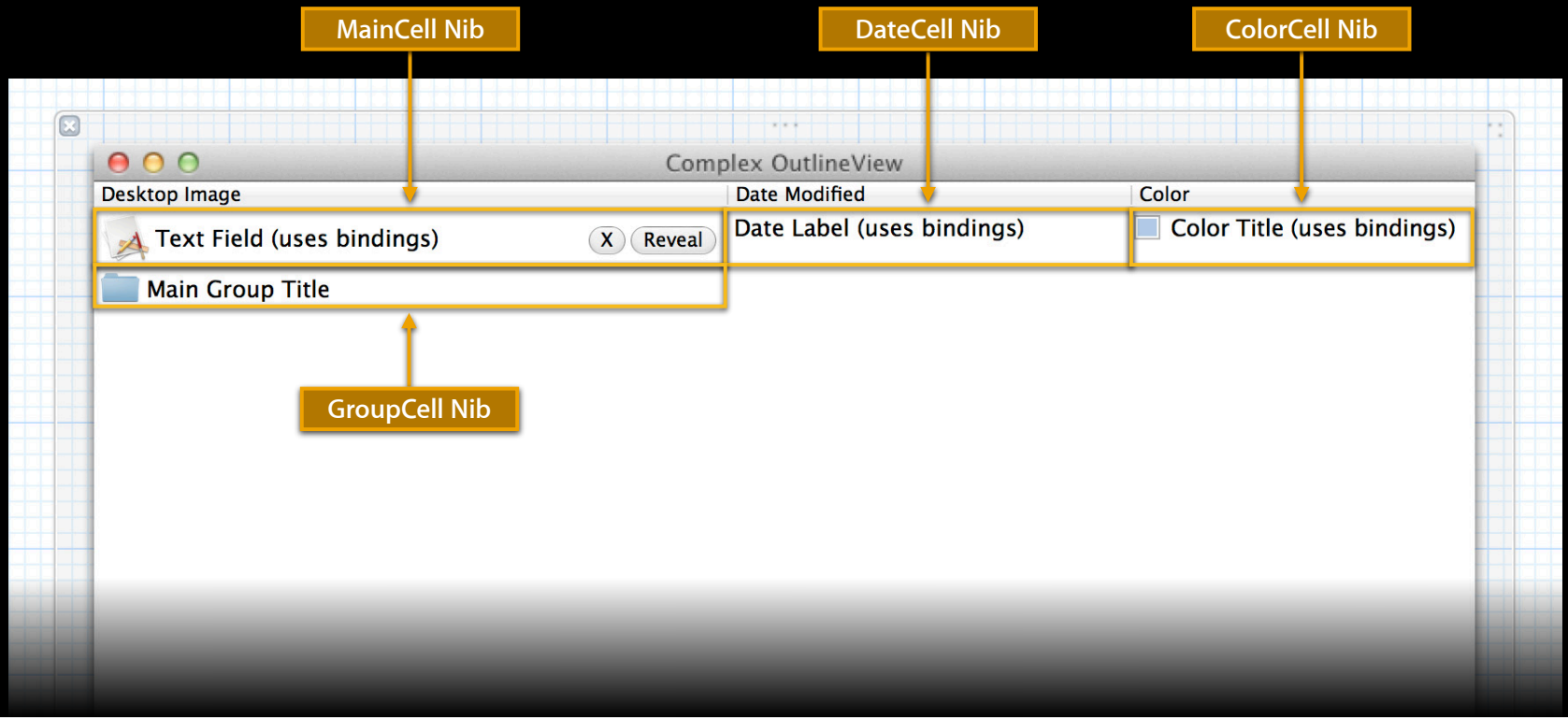
```
@interface NSNib : NSObject <NSCoding> ...

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

@end
```



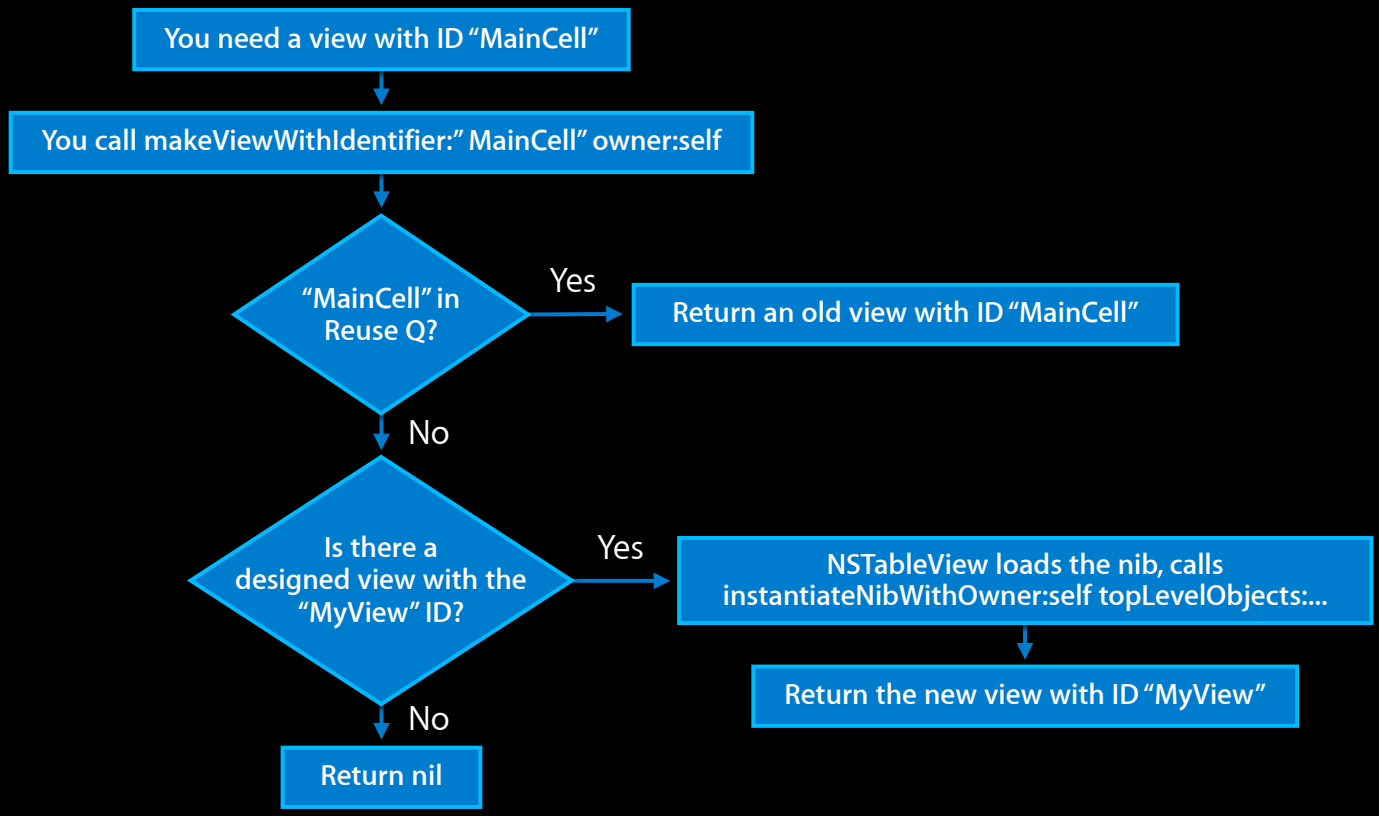
NSTableView Cell Encoding

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

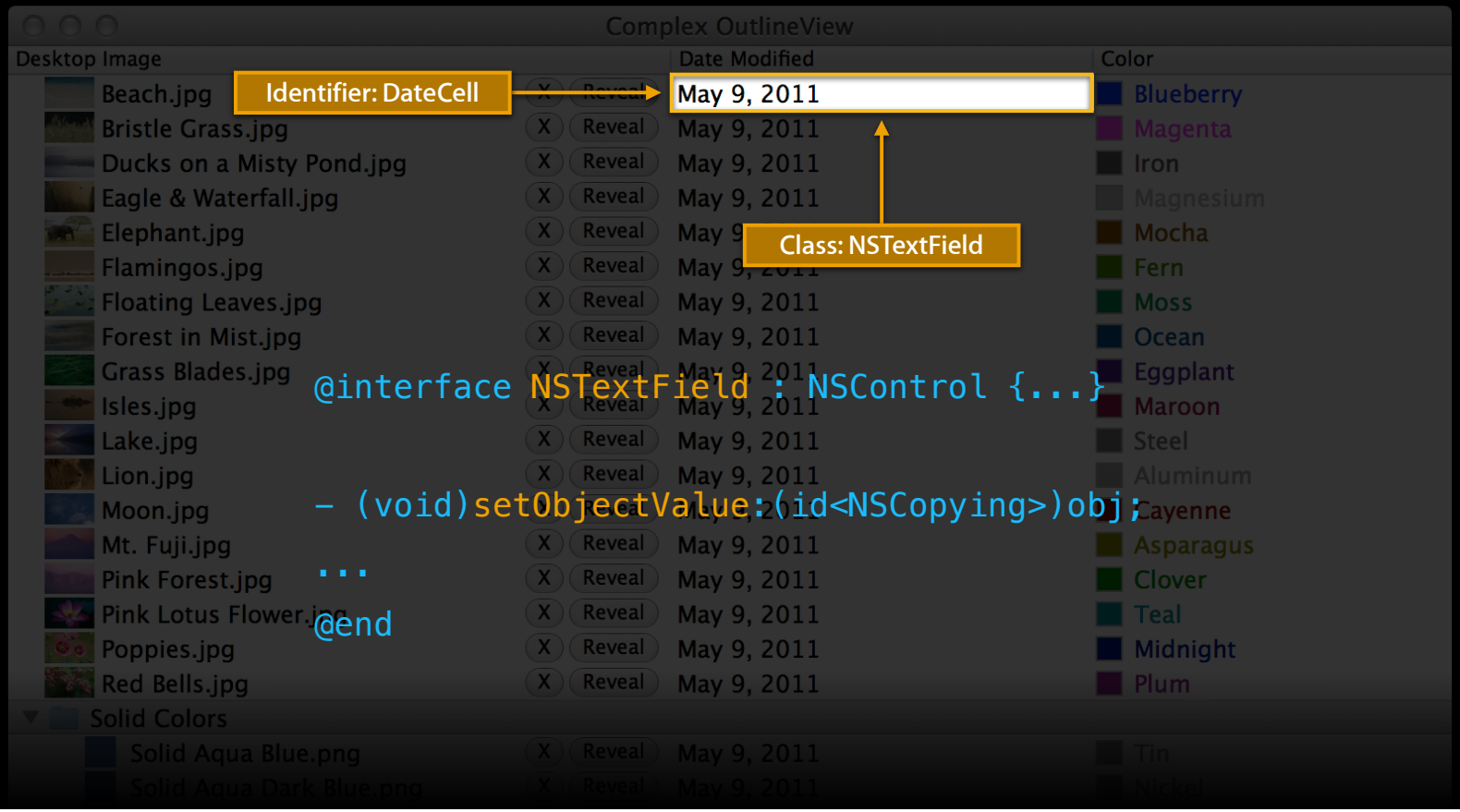
Providing data for a particular cell

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

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

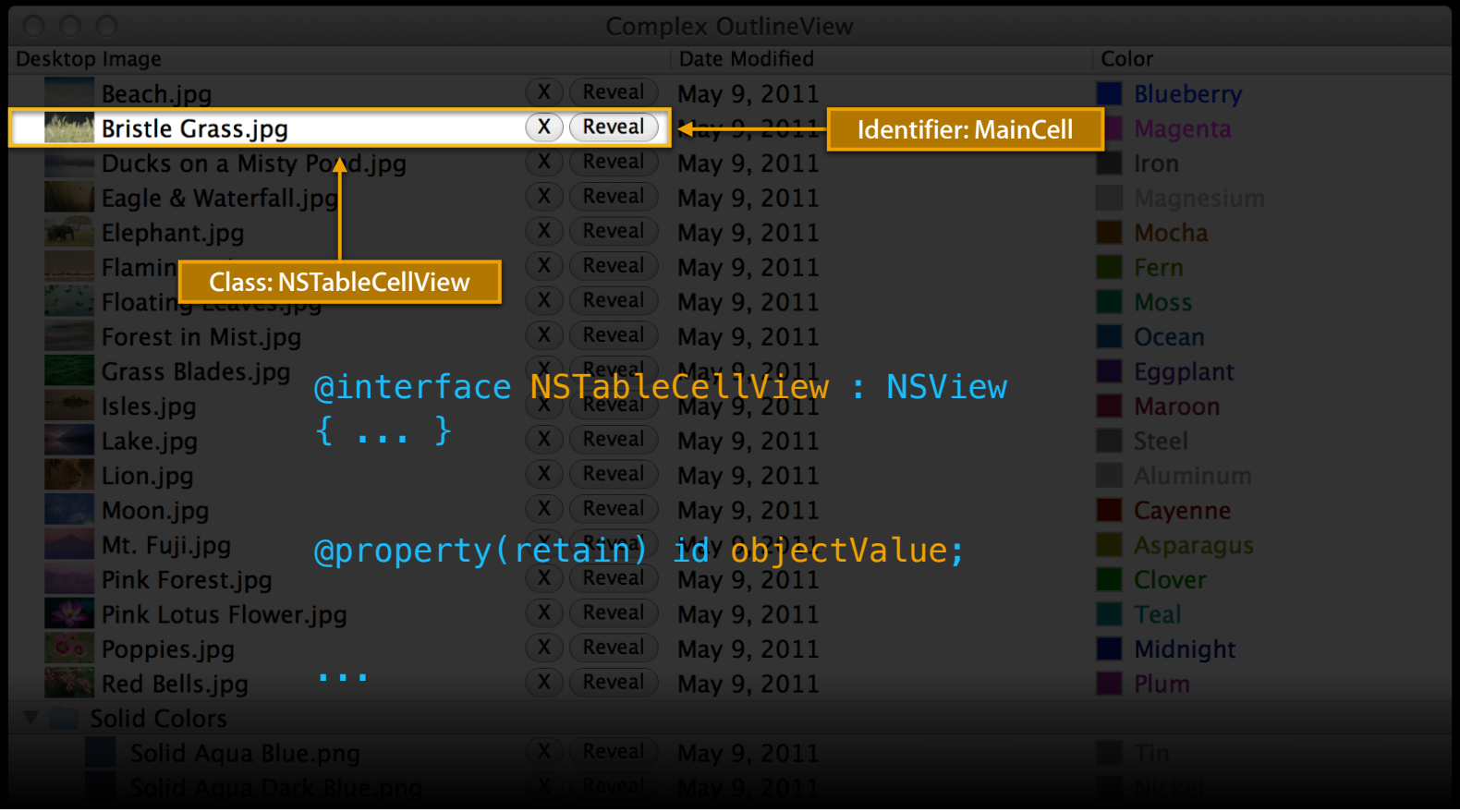
  - Generally, return your model object value

Simple -objectValue

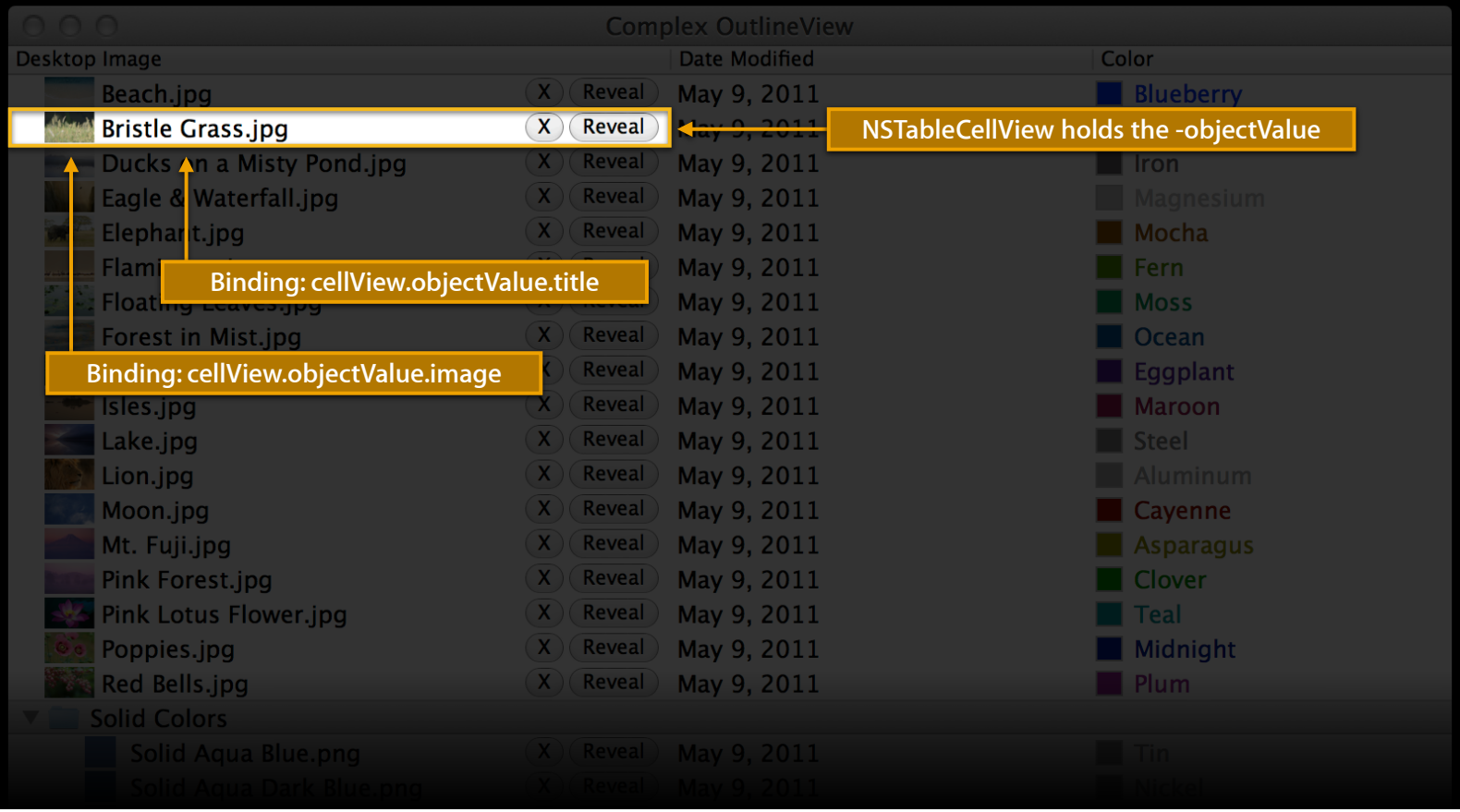



```
@interface NSTextField : NSControl {...}

- (void)setObjectValue:(id<NSCopying>)obj;
...
@end
```

Complex -objectValue

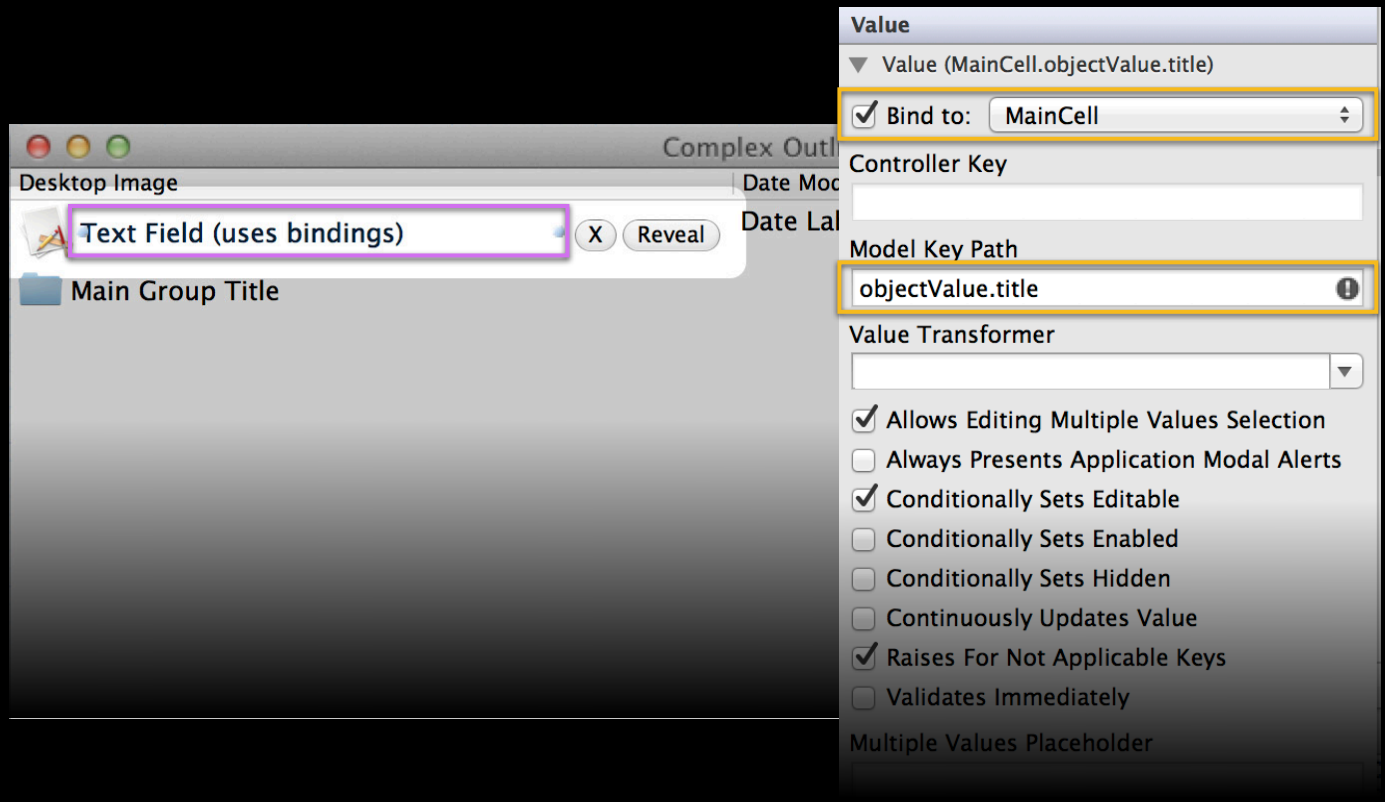


Complex -objectValue



Inside Xcode 4

NSTableCellView has an objectValue binding



Automatic View Loading

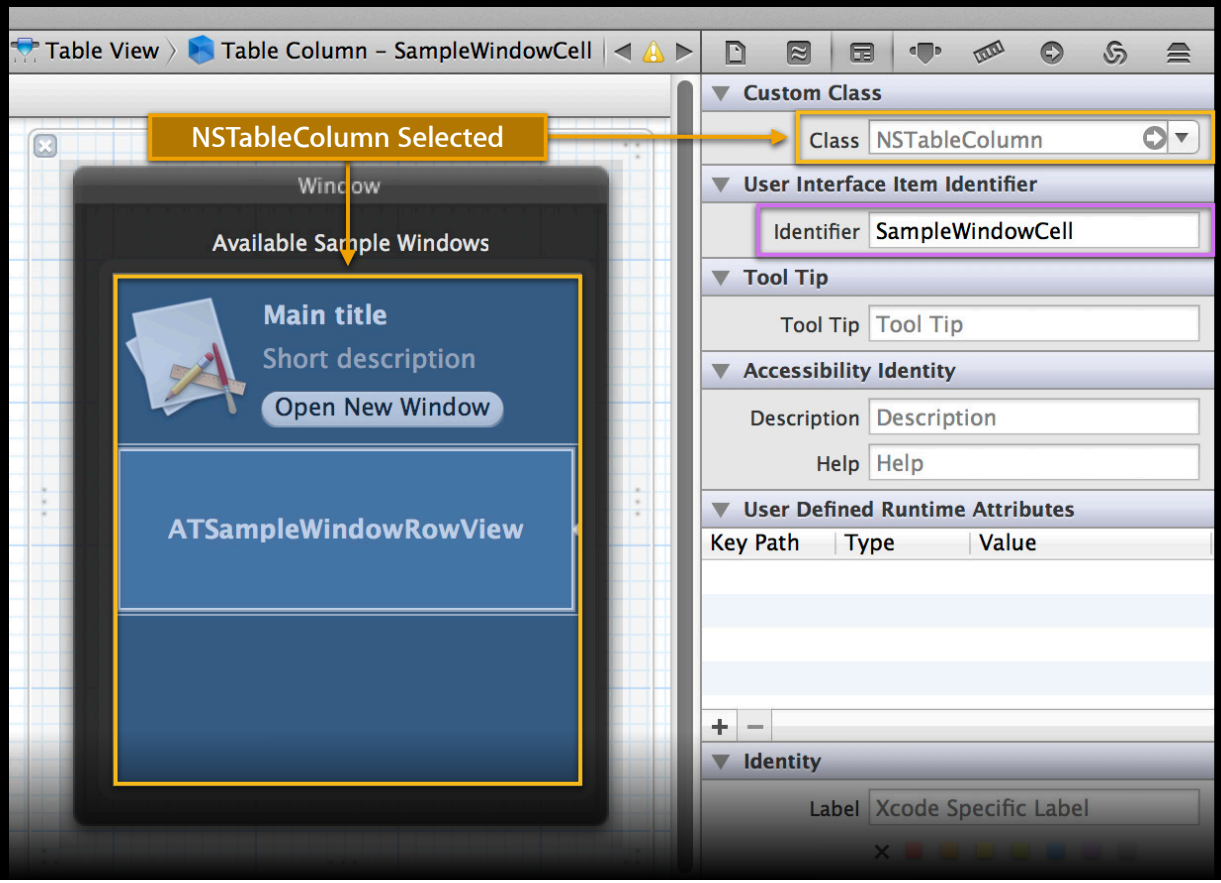
- Optional delegate method

  `tableView:viewForTableColumn:row`

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

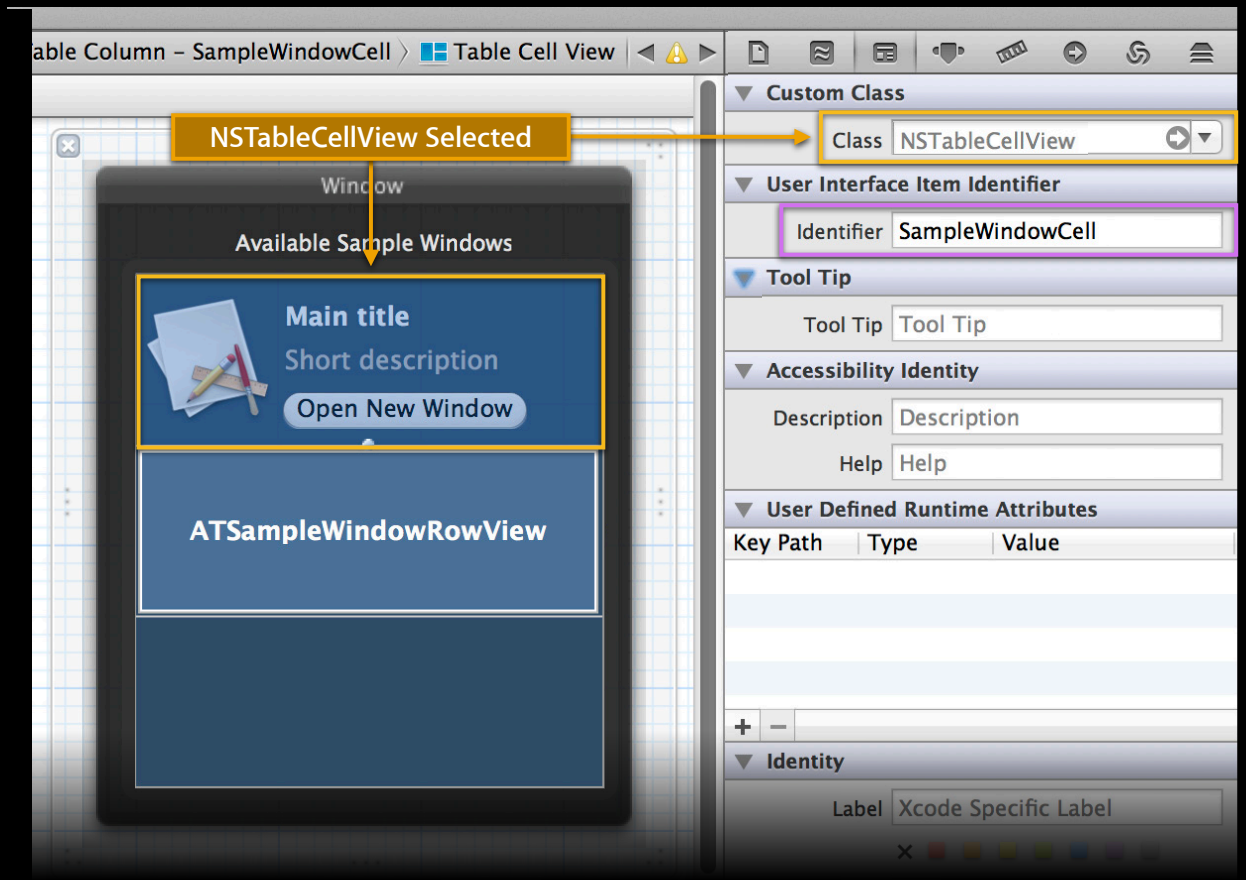
Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib



Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib



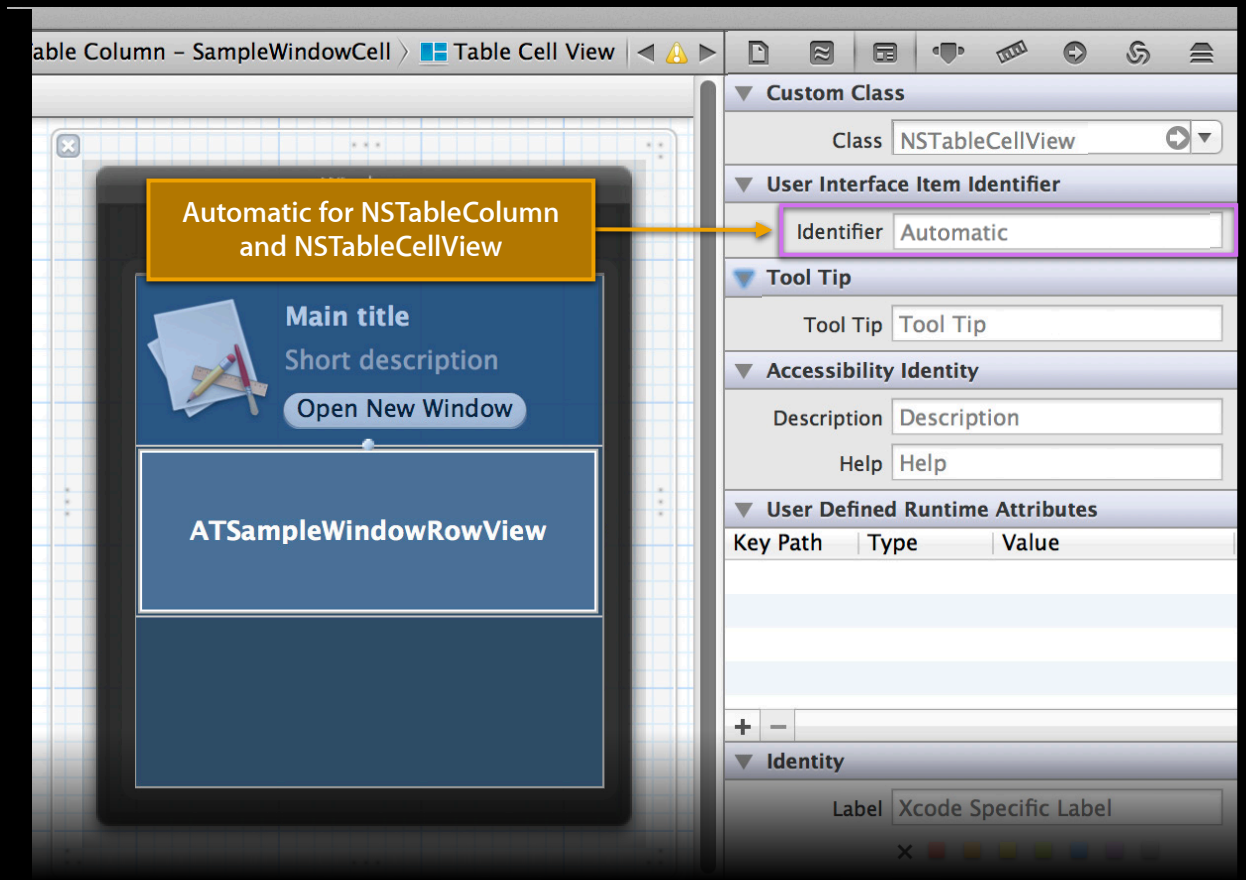
Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib

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

Automatic View Loading

TableViewPlaygroundDemo—MainMenu.xib

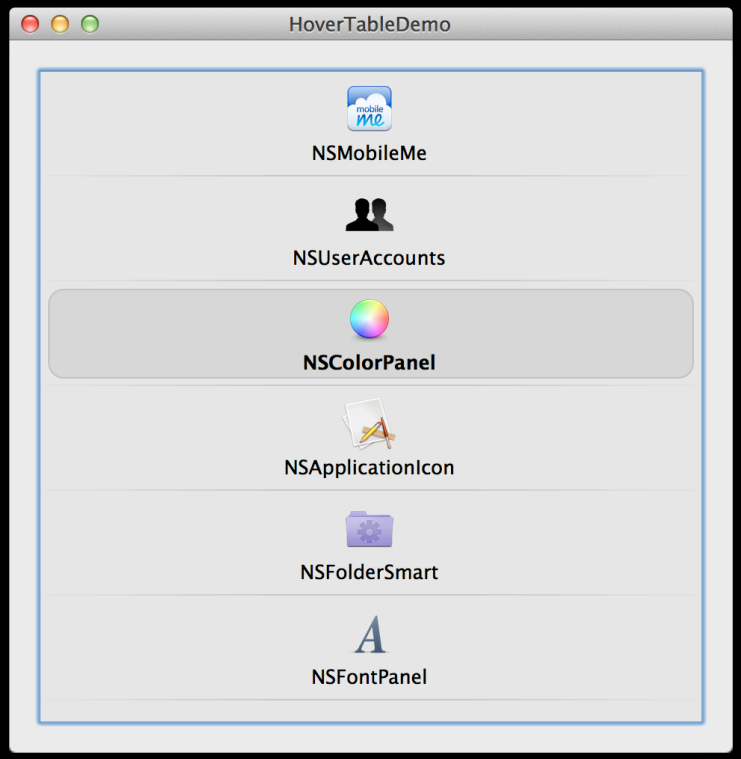


Demo

Customizing in the HoverTableDemo

Custom Row Selection

Subclass NSTableRowView



Custom Row Selection

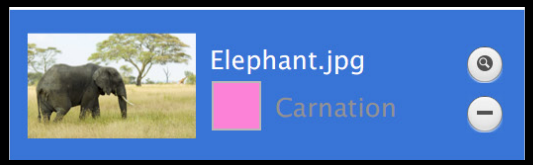
Subclass NSTableRowView


```
- (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];
}
```

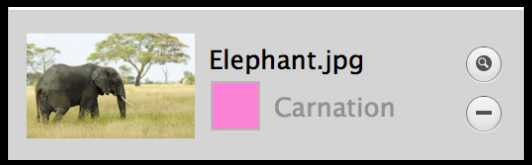
Custom Row Selection

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];
      }
    }];
}
```

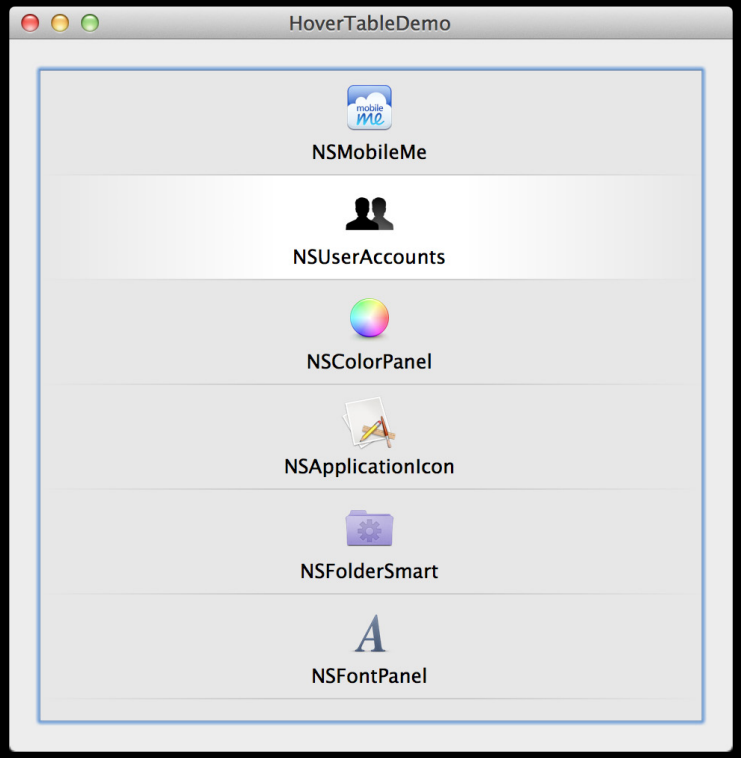
Bold Selected Text

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

NSTableRowView subclass



Hover Effect

NSTableRowView 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];
    }
}
```

Hover Effect

NSTableRowView subclass


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

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

Hover Effect

NSTableRowView 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];
    }
}
```

Hover Effect

NSTableRowView 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];
    }
}
```

Hover Effect

NSTableRowView 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];
    }
}
```

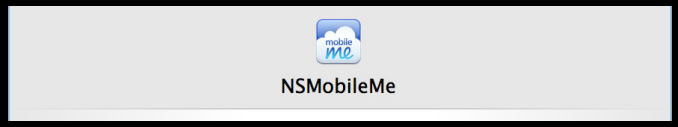
Hover Effect

NSTableRowView 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];
    }
}
```

Custom Separator

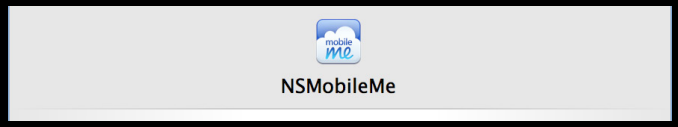
NSTableRowView subclass




```
- (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

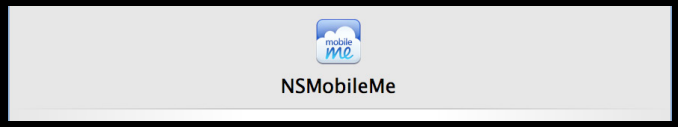
NSTableRowView subclass




```
- (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

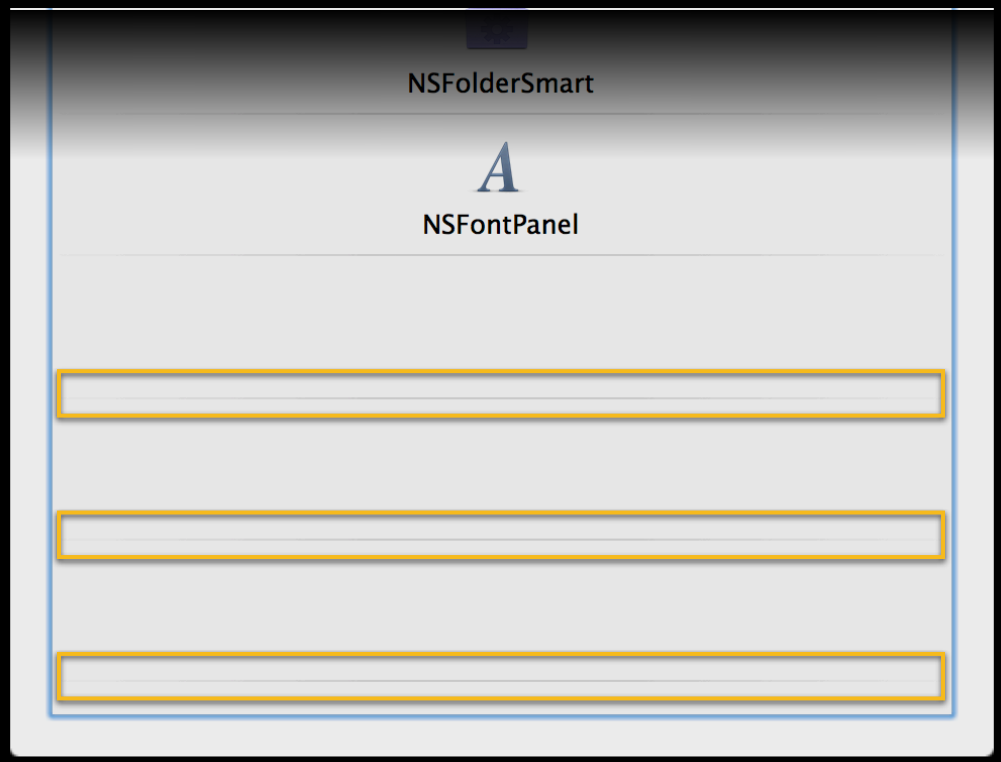
NSTableRowView subclass




```
- (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:NSMakeRect(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:NSMakeRect(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:NSMakeRect(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:NSMakeRect(0, 0, 100, 100)];
  // Setup custom properties on the row view
  return [result autorelease];
}
```

Providing Custom Row Views

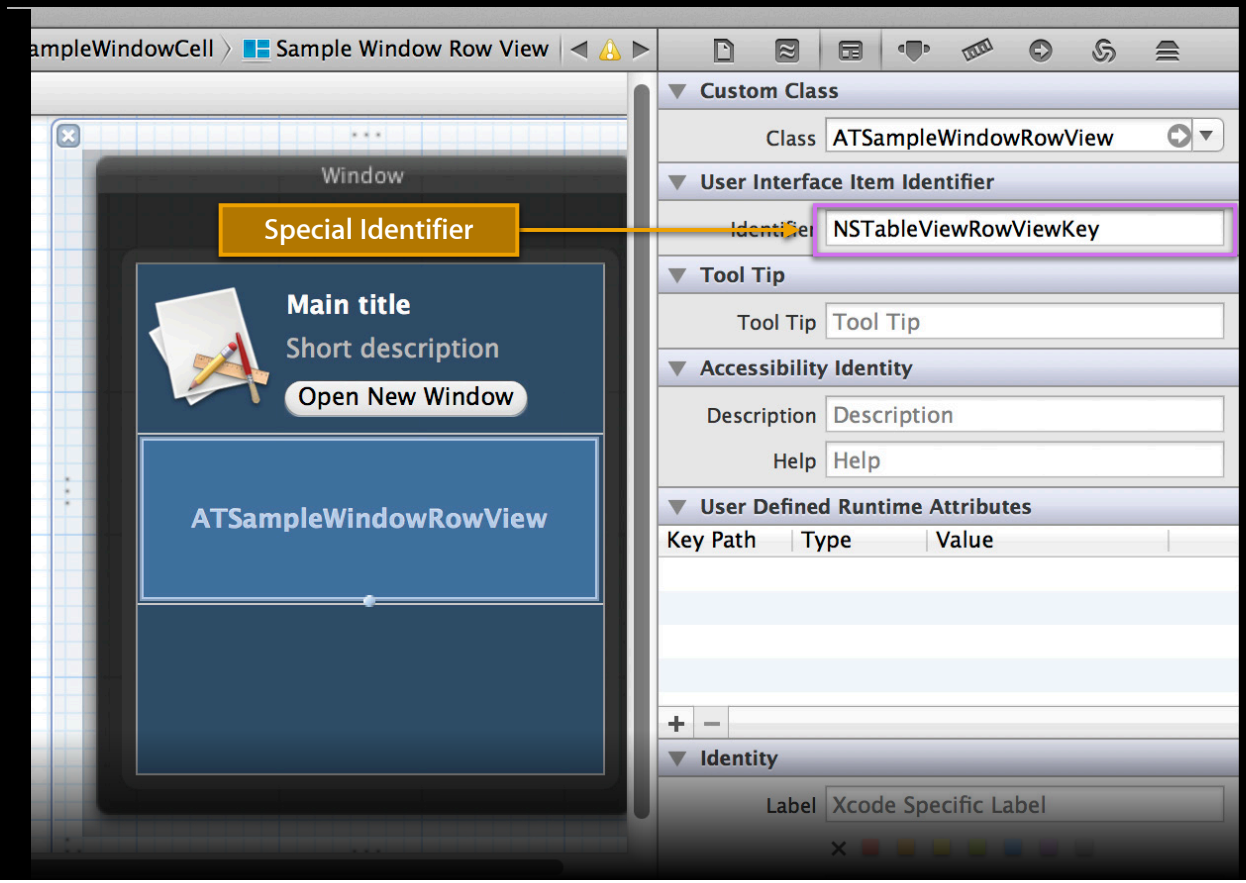
Can also be done at design time

```
- (NSTableRowView *)tableView:(NSTableView *)tableView
                rowViewForRow:(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



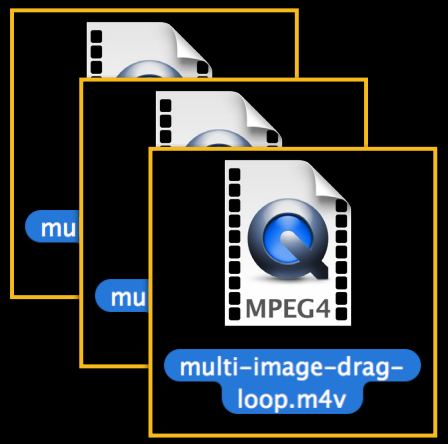
Demo

Drag and drop in TableViewPlayground

Multi-Image Dragging

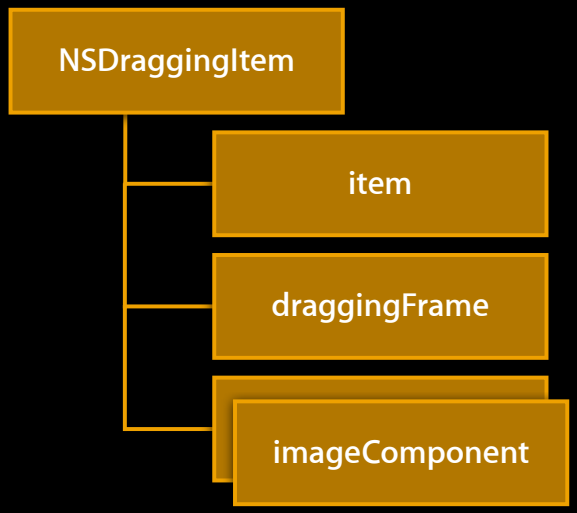
New

- 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];
}
```

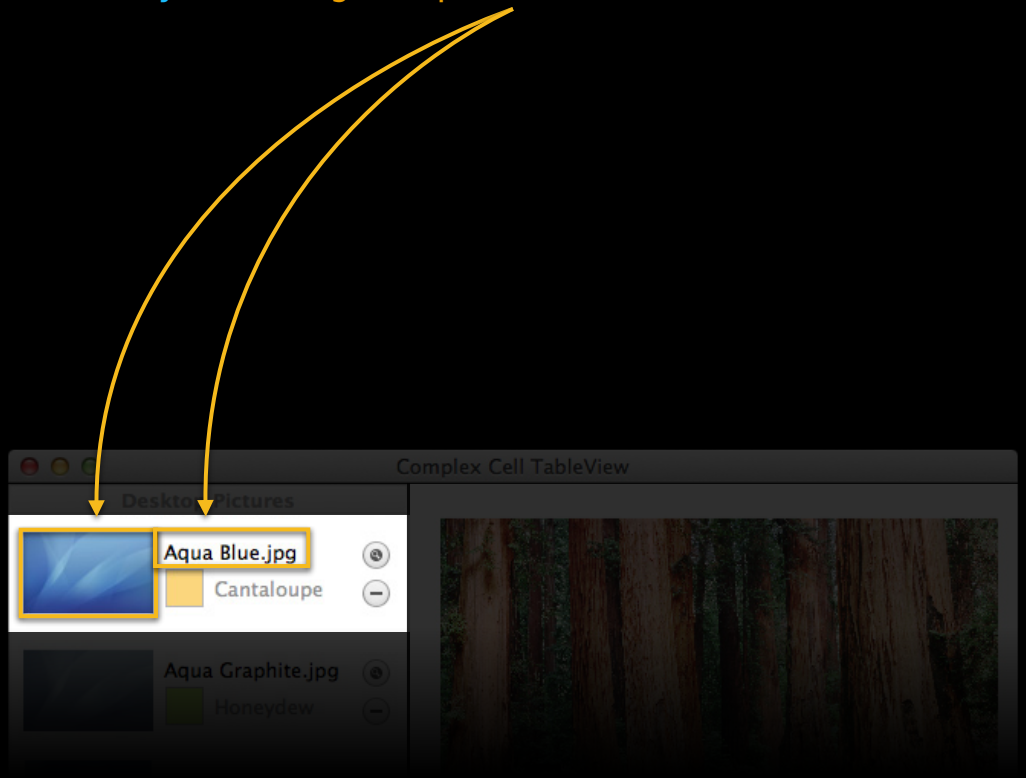
Placing Items on the Pasteboard

Implementing NSPasteboardWriter

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

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

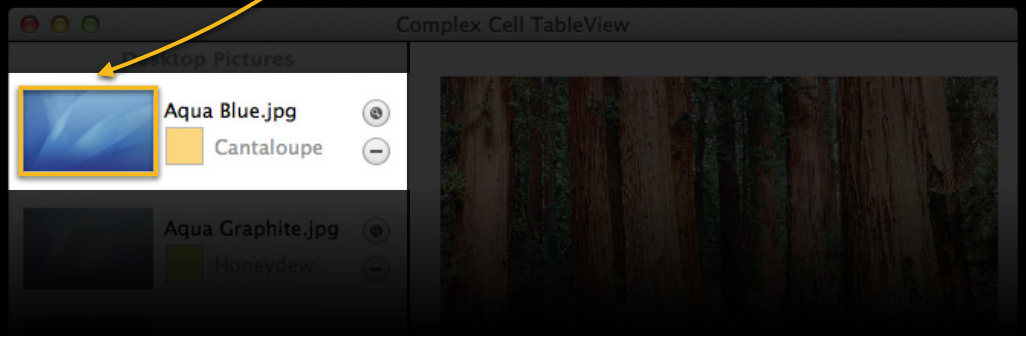
Providing Drag Images

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

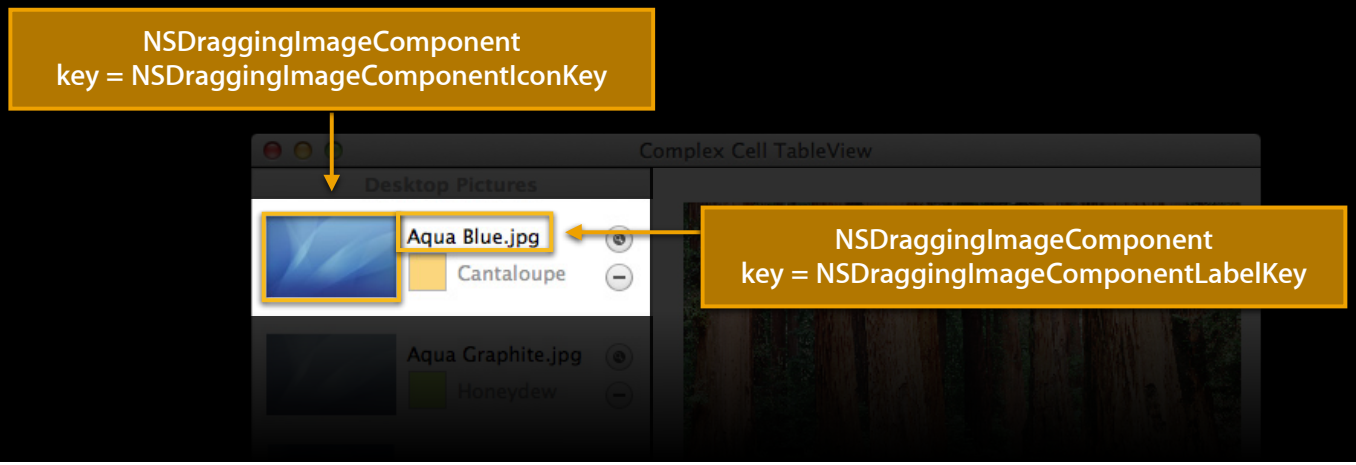


Providing Drag Images

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

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



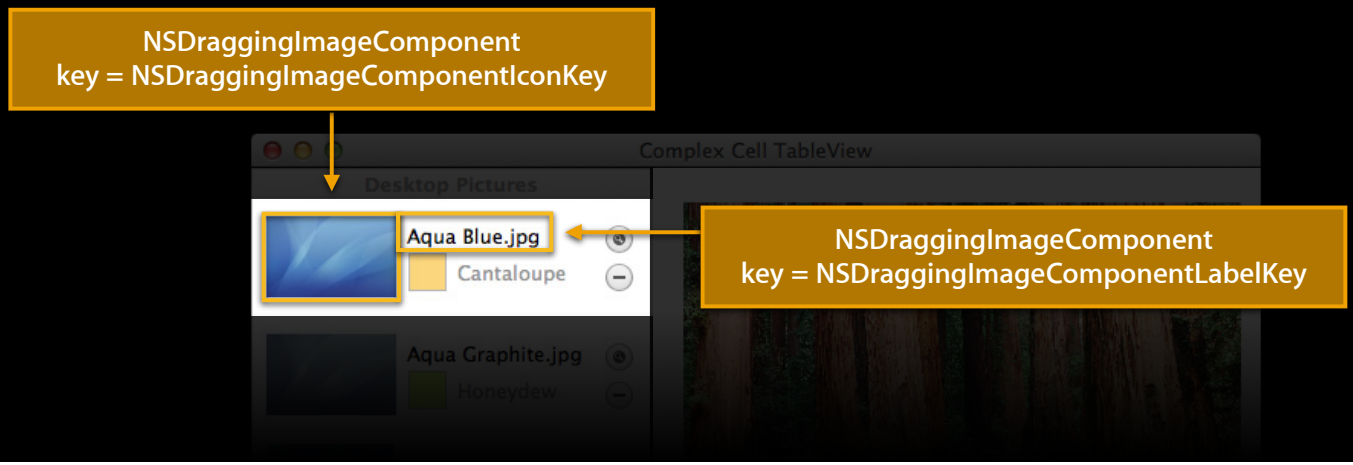
Providing Drag Images



Providing Drag Images

- Automatically provided by NSTableCellView:

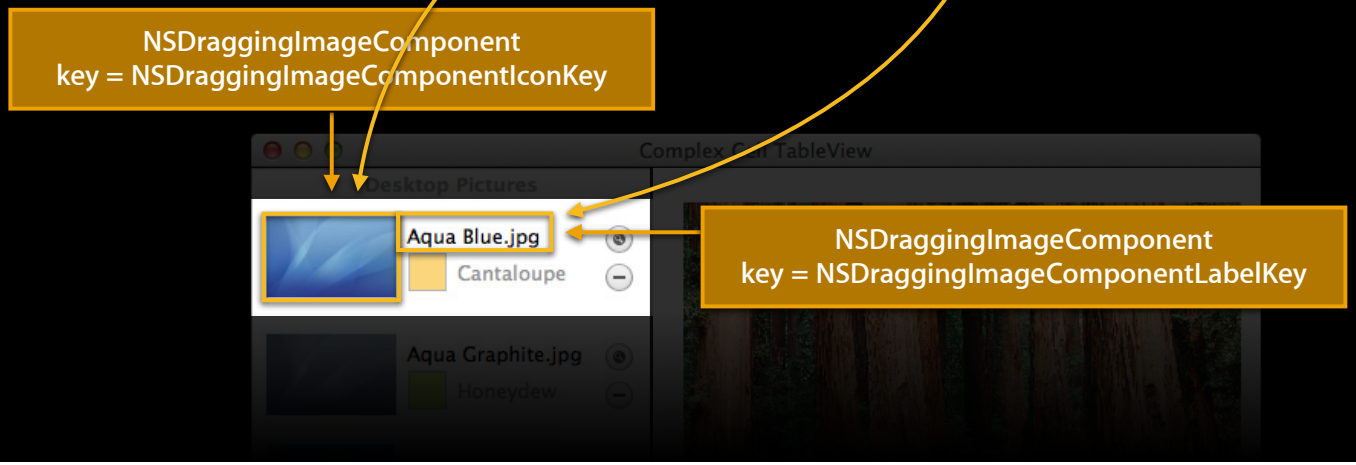
```
@property(retain, readonly) NSArray *draggingImageComponents;
```



Providing Drag Images

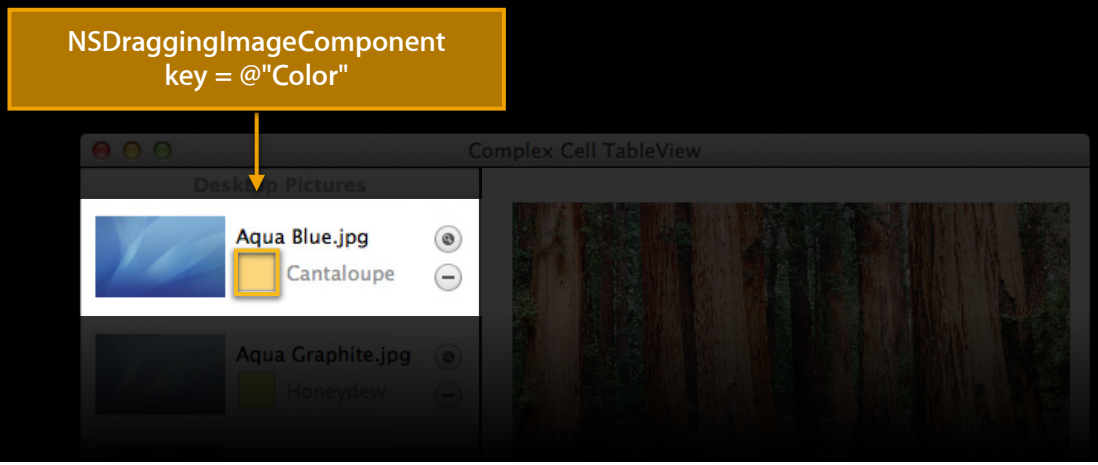
- Automatically provided by NSTableCellView:

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



Providing Drag Images

- How do you add other items?



Providing Drag Images

Subclass NSTableCellView

```
- (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 NSTableCellView

```
- (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 NSTableCellView

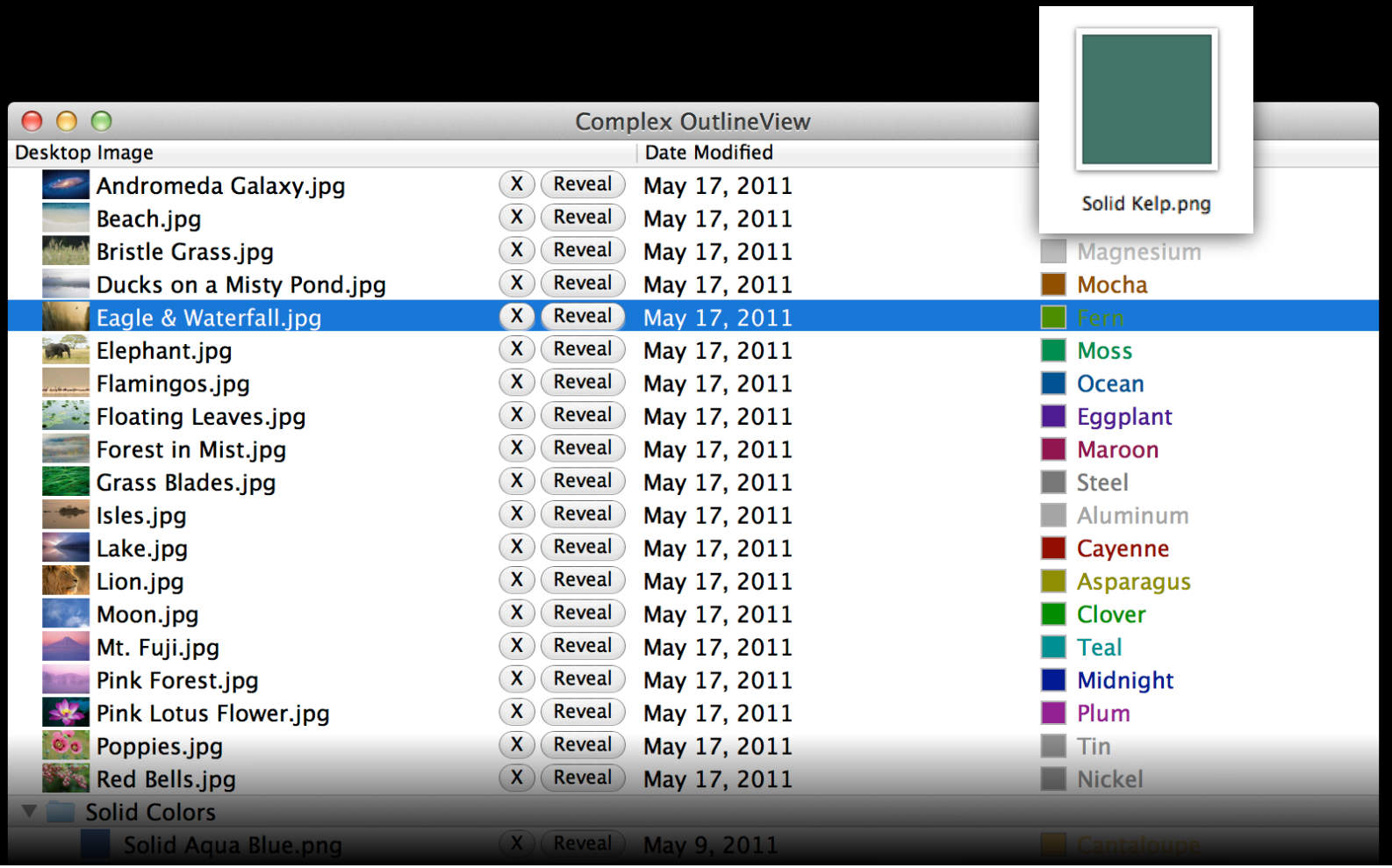
```
- (NSArray *)draggingImageComponents {
  NSMutableArray *result =
                [[[super draggingImageComponents] mutableCopy] autorelease];

  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

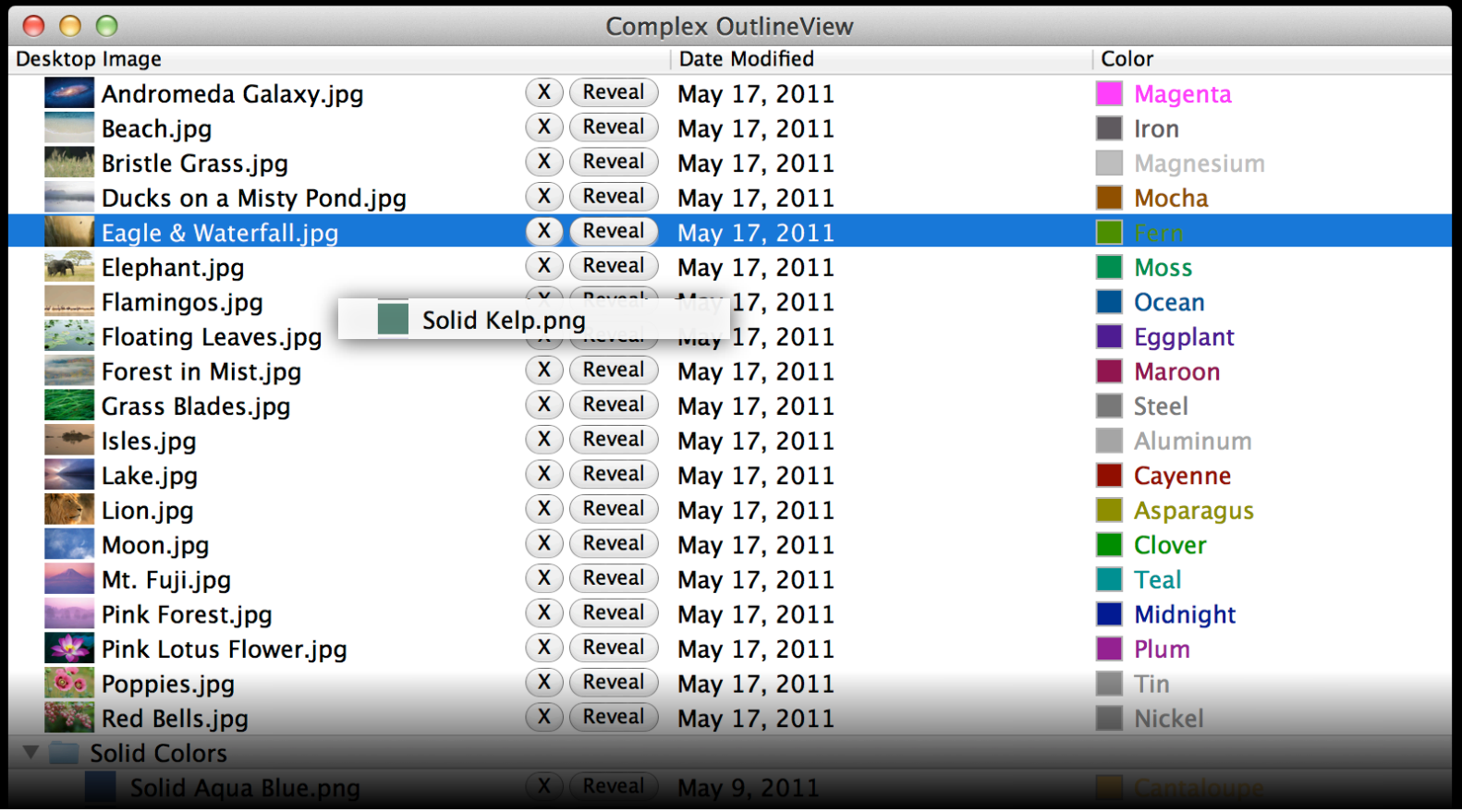
Subclass NSTableCellView

```
- (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



Changing Drag Images



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 {

  ATTableCellView *tableCellView =
       [outlineView makeViewWithIdentifier:@"MainCell" owner:self];
  NSRect cellFrame = NSMakeRect(0, 0,
                        [tableColumn width], [outlineView rowHeight]);
```

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 = cellFrame;
      draggingItem.imageComponentsProvider = ^(void) {
           tableCellView.objectValue = draggingItem.item;
           tableCellView.frame = cellFrame;
           return [tableCellView 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 = cellFrame;
      draggingItem.imageComponentsProvider = ^(void) {
           tableCellView.objectValue = draggingItem.item;
           tableCellView.frame = cellFrame;
           return [tableCellView 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 = cellFrame;
      draggingItem.imageComponentsProvider = ^(void) {
           tableCellView.objectValue = draggingItem.item;
           tableCellView.frame = cellFrame;
           return [tableCellView draggingImageComponents];
      };
}
```

Changing Drag Images

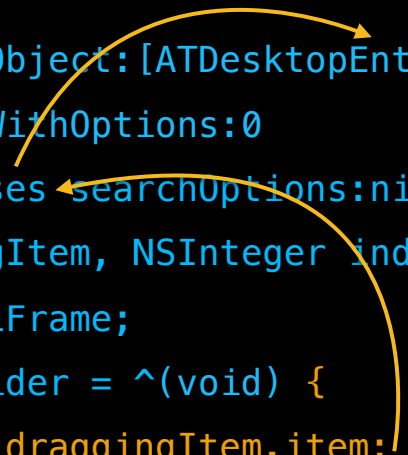
Enumerate the source dragging items

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

Changing Drag Images

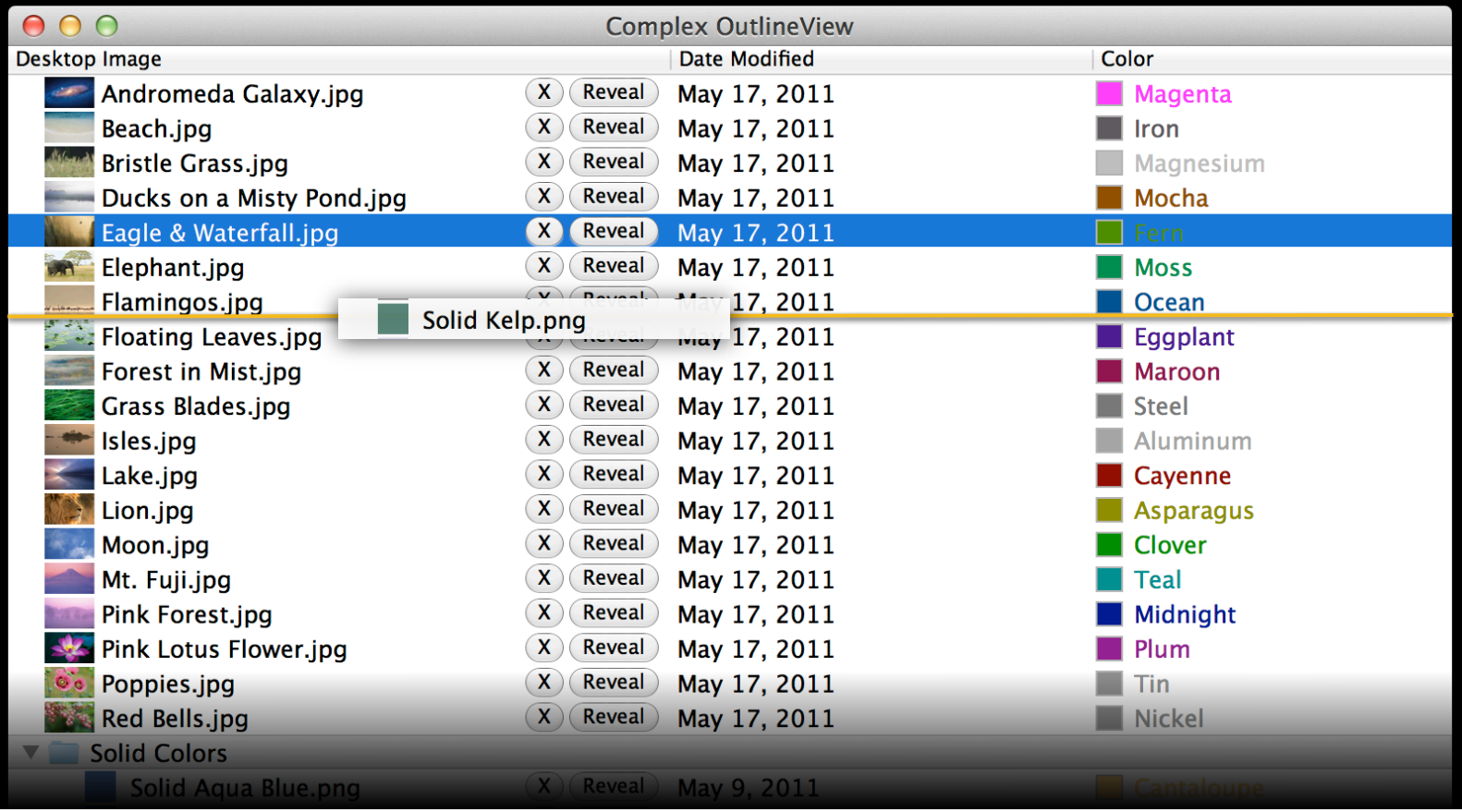
Enumerate the source dragging items

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



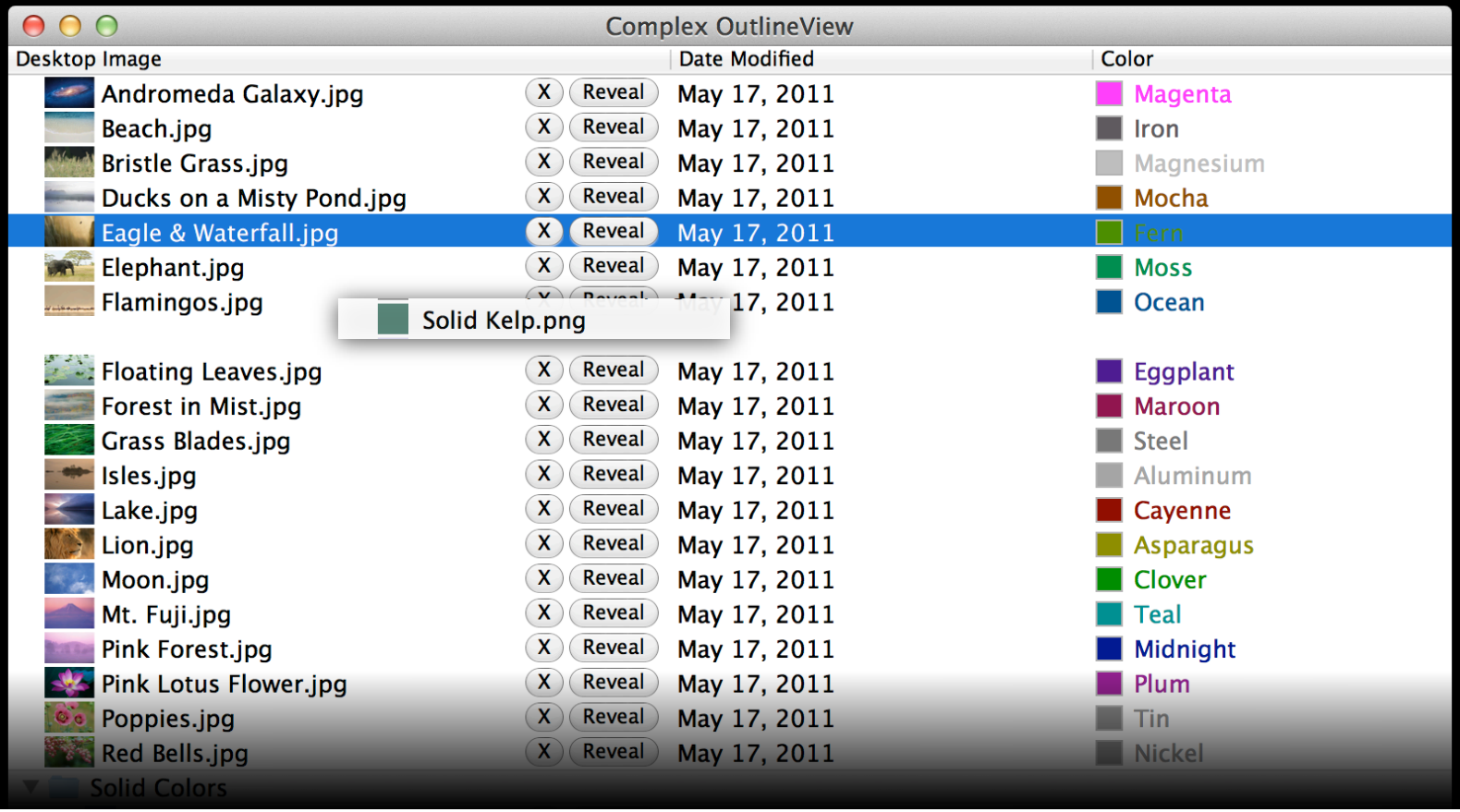
Accepting a Drop

Along with an animation



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];
  NSIndexSet *indexes = [NSIndexSet 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];
  NSIndexSet *indexes = [NSIndexSet indexSetWith
  [_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];
  NSIndexSet *indexes = [NSIndexSet indexSetWithIndex:insertionIndex];
  [_                    tItemsAtIndexes: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 inse                                    dex:insertionIndex];
  NSIndexSet *indexes =                                    tWithIndex: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];
  NSIndexSet *indexes = [NSIndexSet indexSetWithIndex:insertionIndex];
  [_outlineView insertItemsAtIndexes:indexes
                            inParent:target
                       withAnimation:NSTableViewAnimationEffectGap];

  NSInteger row = [_outlineVie
  draggingItem.draggingFrame =                                  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];
  NSIndexSet *indexes = [NSIndexSet indexSetWithIndex:insertionIndex];
  [_outlineView insertItemsAtIndexes:indexes
                            inParent:target
                       withAnimation:NSTableViewA
```

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

```
  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];
  NSIndexSet *indexes = [NSIndexSet 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];
  NSIndexSet *indexes = [NSIndexSet indexSetWithIndex:insertionIndex];
  [_outlineView insertItemsAtIndexes:indexes
                            inParent:target
                       withAnimation:NSTableViewAnimationEffectGap];

  NSInte                                  w 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;
```

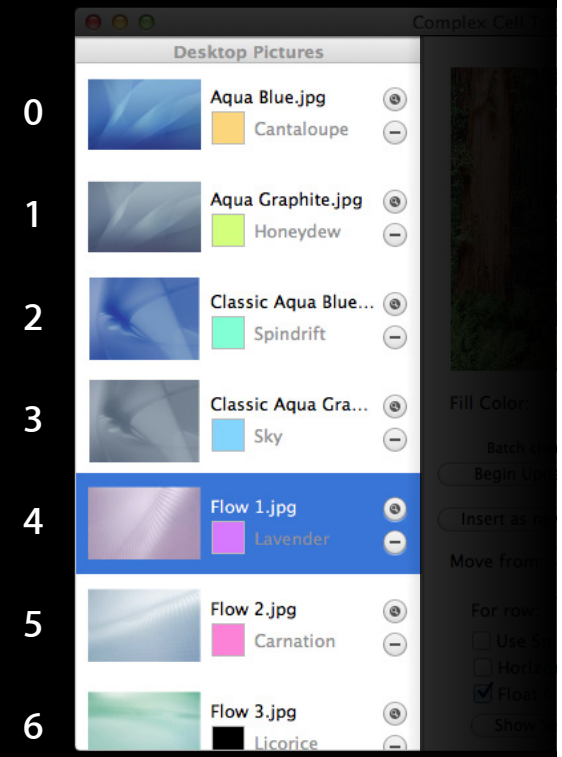
Insertion

Similar to NSMutableArray

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

New Row ONE

New Row THREE



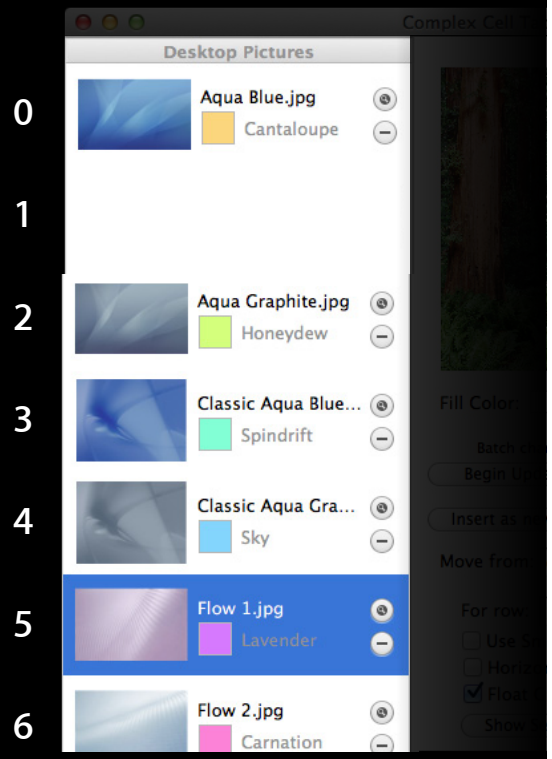
Insertion

Similar to NSMutableArray

```
– (void)insertRowsAtIndexes: 1, 3 withAnimation: ...
```

New Row ONE

New Row THREE

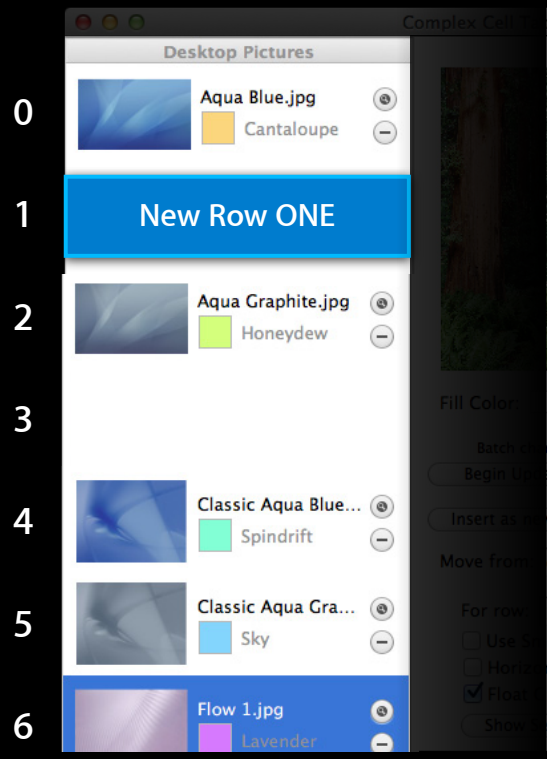


Insertion

Similar to NSMutableArray

```
– (void)insertRowsAtIndexes: 1, 3 withAnimation: ...
```

New Row THREE



Batch Updates

Better for performance

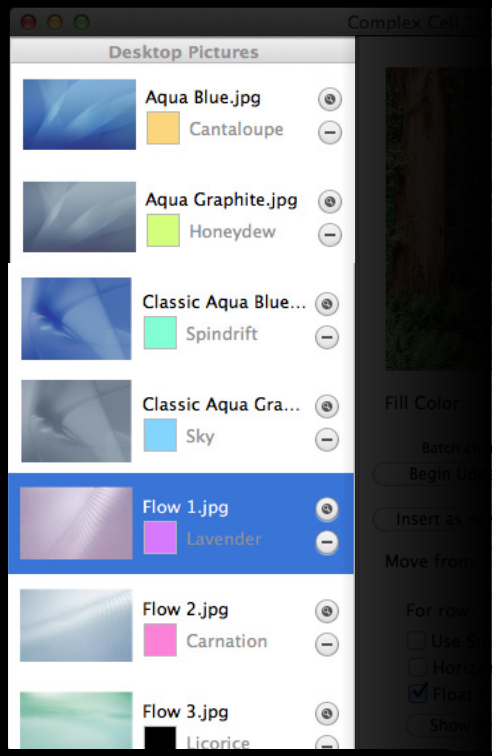
```
[table beginUpdates];

[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

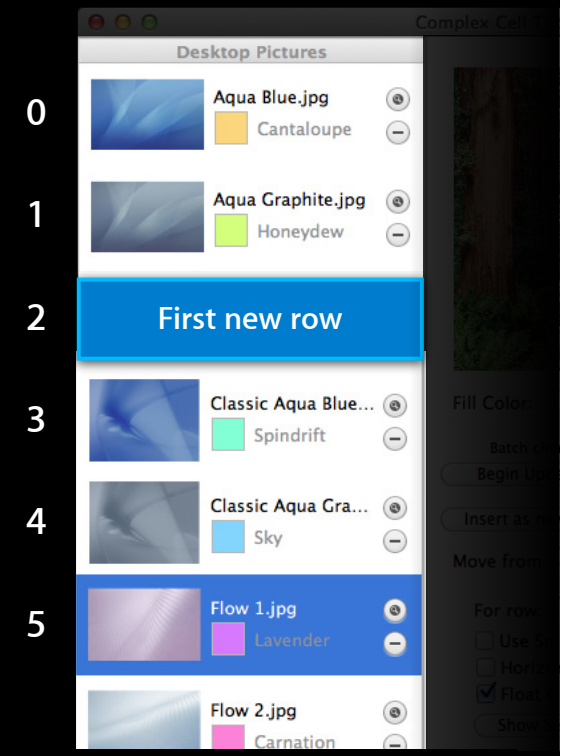


Batch Updates

Similar to NSMutableArray

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

Second new row

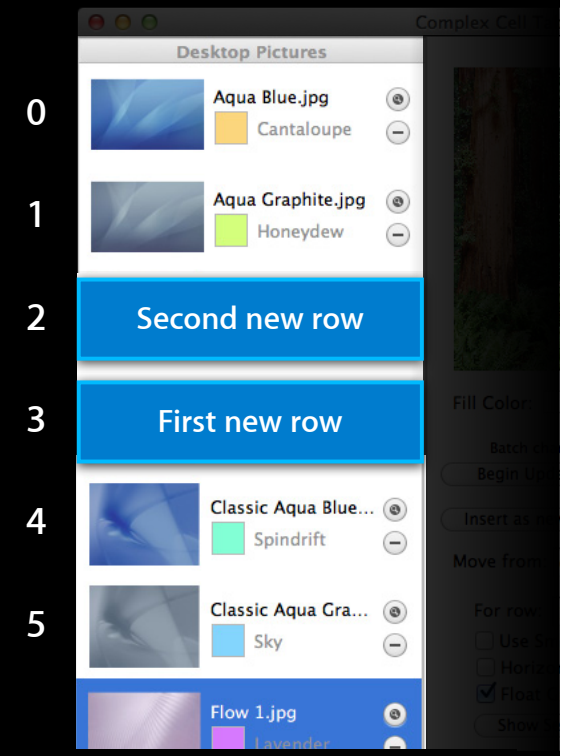


Batch Updates

Similar to NSMutableArray

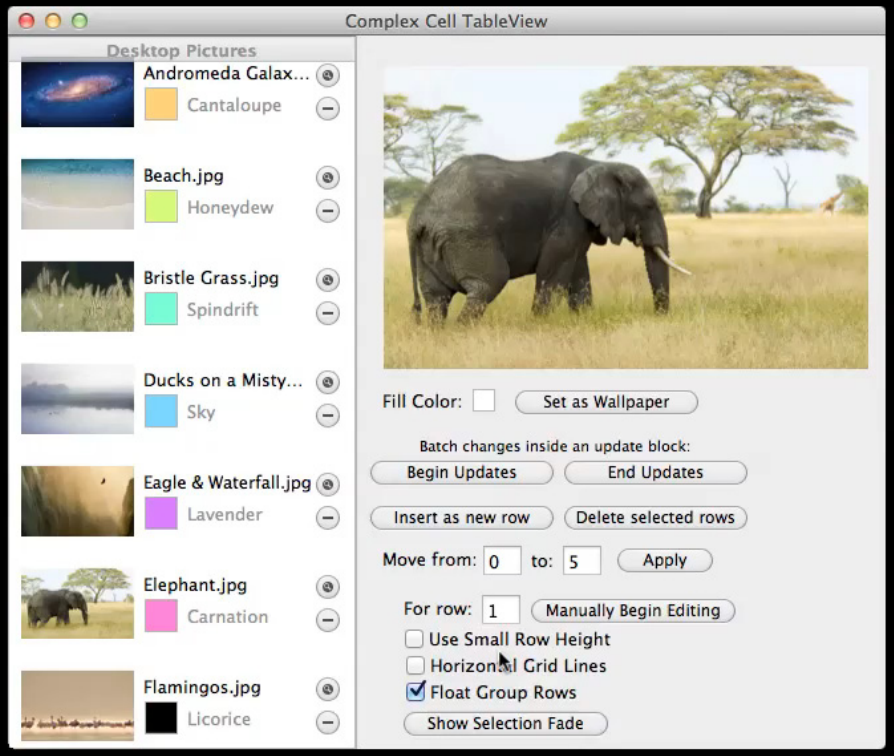
```
[table beginUpdates];
[table insertRowsAtIndexes:2 ... ];
[table insertRowsAtIndexes:2 ... ];
[table insertRowsAtIndexes:2 ... ];
[table endUpdates];
```

Third new row



Row Height Animations

As seen in TableViewPlayground



Row Height Animations

As seen in TableViewPlayground

```
NSIndexSet *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


WWDC2011