



Grid Lanes

Tim Finch

MS Access Forms User Control
written in VBA Classes and GDI Calls

AEU Online May 2026

www.timfinch.co.uk

@timbald on Discord



GridLanes: VBA 'User Control'

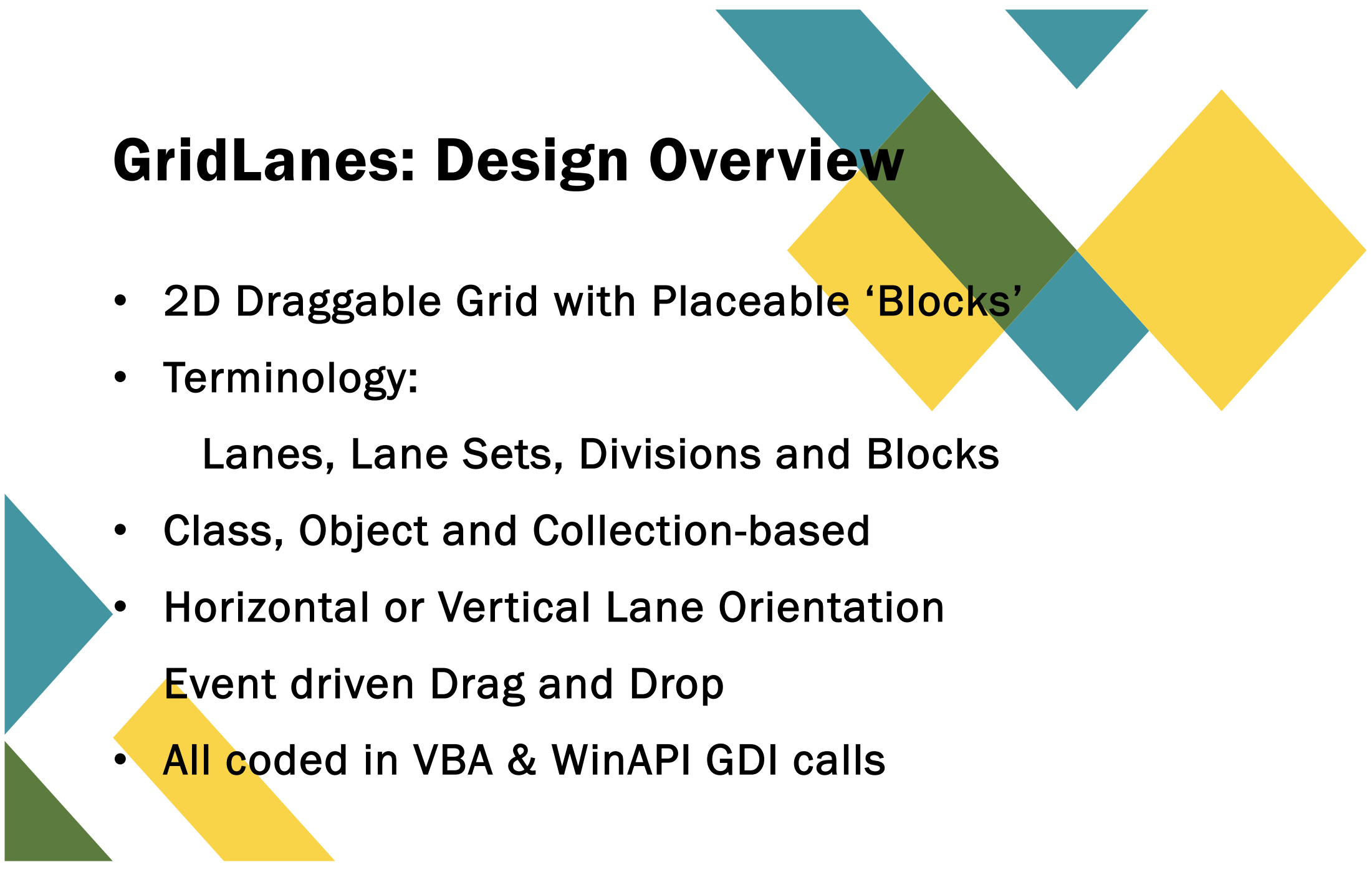
Grid Lanes

03/10/2025

Refresh

Close

	Emelda Laroute	Jane Pieterston	Mike Smith	Ravi Iscarri
1st Slot	TAILOR, Mr Yusef EH32 0FF	GRANT, Ms Abby EH11 1AF	GATFIELD, Ms Cindy EH51 9BH	HART, Mr Gregory EH5 2PL
2nd Slot	ESKDALE, Mr Anthony EH3 6HD	YOLANDER, Mr Aaron EH15 2EZ	LEADBETTER, Mr Benjamin EH27 8AF	PEAKE, Ms Caroline EH14 1DF
3rd Slot	VERNE, Ms Adele EH45 8EW	JENKINSON, Miss Danielle EH15 3PA	EMERSON, Ms Tia EH11 1PP	
4th Slot	ANDERSON, Mrs Fay EH54 8FB	WAKEMAN, Mr Ashley EH16 4AP	NETHERTON, Mr Edwin EH55 8XA	
5th Slot				

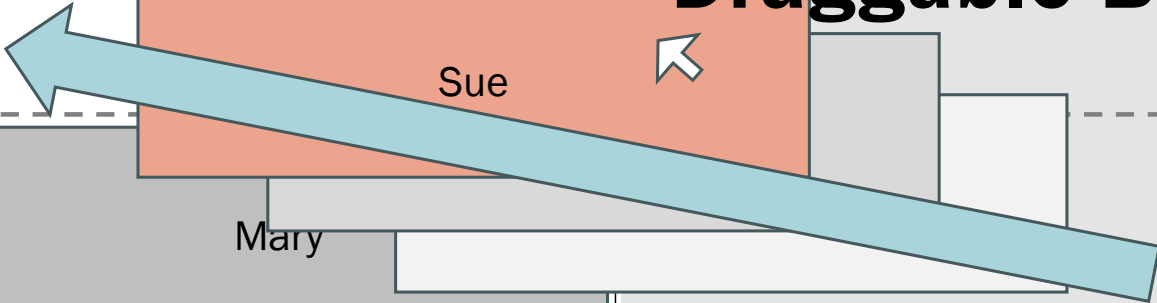


GridLanes: Design Overview

- 2D Draggable Grid with Placeable 'Blocks'
- Terminology:
 - Lanes, Lane Sets, Divisions and Blocks
- Class, Object and Collection-based
- Horizontal or Vertical Lane Orientation
- Event driven Drag and Drop
- All coded in VBA & WinAPI GDI calls

	Answer	Question
1st		Sue
2nd	Mary	
3rd		Pete

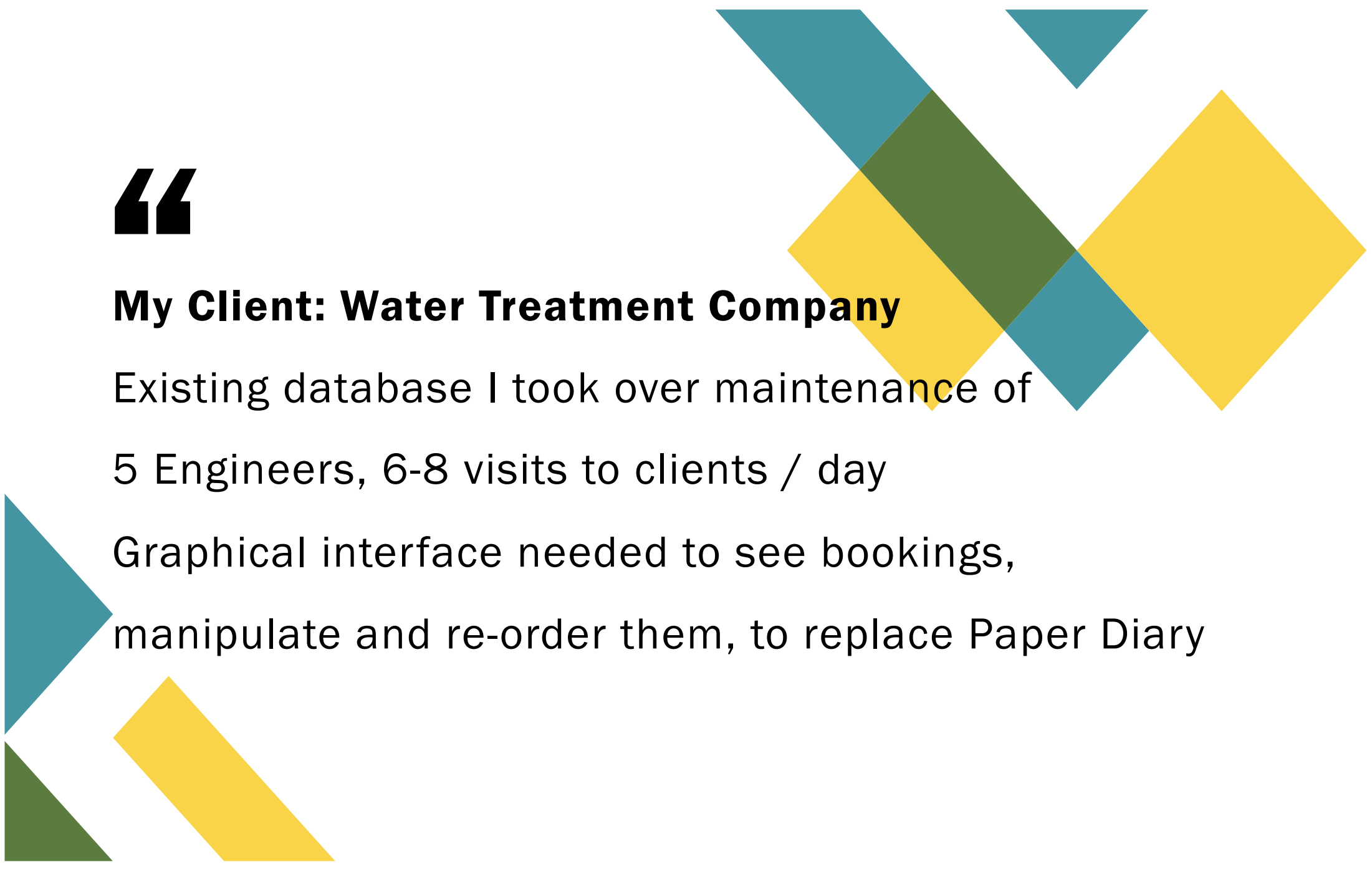
Draggable Blocks





Many thanks to the inspiration of Aleksander Wojtasz

Gantt Project demoed at AUG Europe
Intro to GDI programming in VBA.
Grid Lanes wouldn't exist without him
Do watch his videos [here](#) and [here](#)



“

My Client: Water Treatment Company

Existing database I took over maintenance of

5 Engineers, 6-8 visits to clients / day

Graphical interface needed to see bookings,

manipulate and re-order them, to replace Paper Diary

GridLanes in use: Engineer Bookings and Edge control map grid

Bookings Grid

Thu 02 Oct 2025

Switch to Engineer Worksheet Screen

See all points for today

Refresh

Close

	Andrew Smith 07 hr 22 m ✓ RONs OK	Greg Chambers 05 hr 18 m ✓ RONs OK	Lucy Smith 06 hr 26 m ✓ RONs OK	Staff Holiday
1st Slot	Mr Bruce Hagburn 07:00	Mr Simon Rigby 07:00 am	Mr Woodrow Hulse 08:30	
2nd Slot	Mr Simon Rigby 07:30	Mr Woodrow Hulse 08:30 late am	Mr Woodrow Hulse 09:30	
3rd Slot	Mr Simon Rigby 08:30	Mr Woodrow Hulse 09:30 PM	Mr Woodrow Hulse 10:30	
4th Slot	Mr Simon Rigby 09:30		Mr Woodrow Hulse 11:30	
5th Slot	Mr Simon Rigby 10:30		Mr Woodrow Hulse 12:30	
6th Slot			Mr Woodrow Hulse 13:30	
7th Slot	** End of Day 14:30	** End of Day 15:30	** End of Day 16:30	
8th Slot				

Location: South on Thu, 02 October 2025 Engineer's Route

Driving time For This Engineer: 3 h 46 m

TA11 6SB, Bancombe Rd, Somerton

SP6 2DP, North Charford, Breamore, For

SP6 2DP, North Charford, Breamore, For

SP6 2DJ, Breamore, Fordingbridge

SP6 2DN, Breamore, Fordingbridge

BH21 7HR, Hinton Martell, Wimborne

BH15 3DG, Bright Rd, Oakdale, Poole

TA11 6SB, Bancombe Rd, Somerton

Add destination

Options

Send directions to your phone

Copy link

via A303

Fastest route now due to traffic conditions

3 hr 35 min

142 miles

Details

Preview

Map data ©2025 Google

United Kingdom

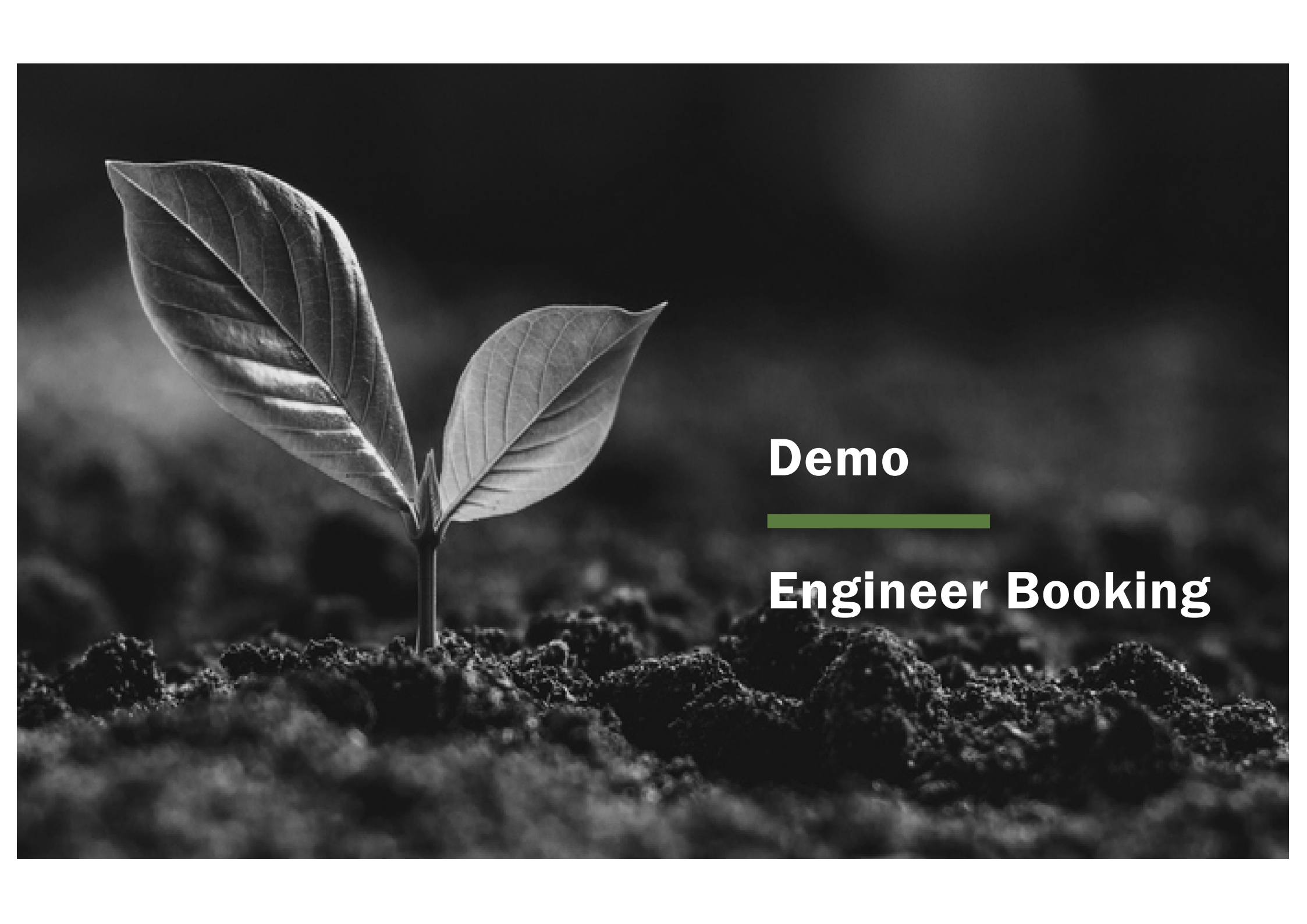
Terms

Privacy

20 km

Gridlanes: A Project In Progress

- Started by rewriting Alek's' code to make it more readable
- Initially used arrays in the Class to hold Block & Lane data
- Almost fully adapted to solely using Objects/Collections
- Spring '26: Migration to using Claude Code to help develop it
 - Immediately saw bugs to fix
 - Conversion to full object orientation
 - Multi block 'Displacement'
 - Edge Resize Feature
 - Cascading Properties (yet to implement)
 - Style Sheets (yet to implement)



Demo

Engineer Booking

Design & VBA Walkthrough

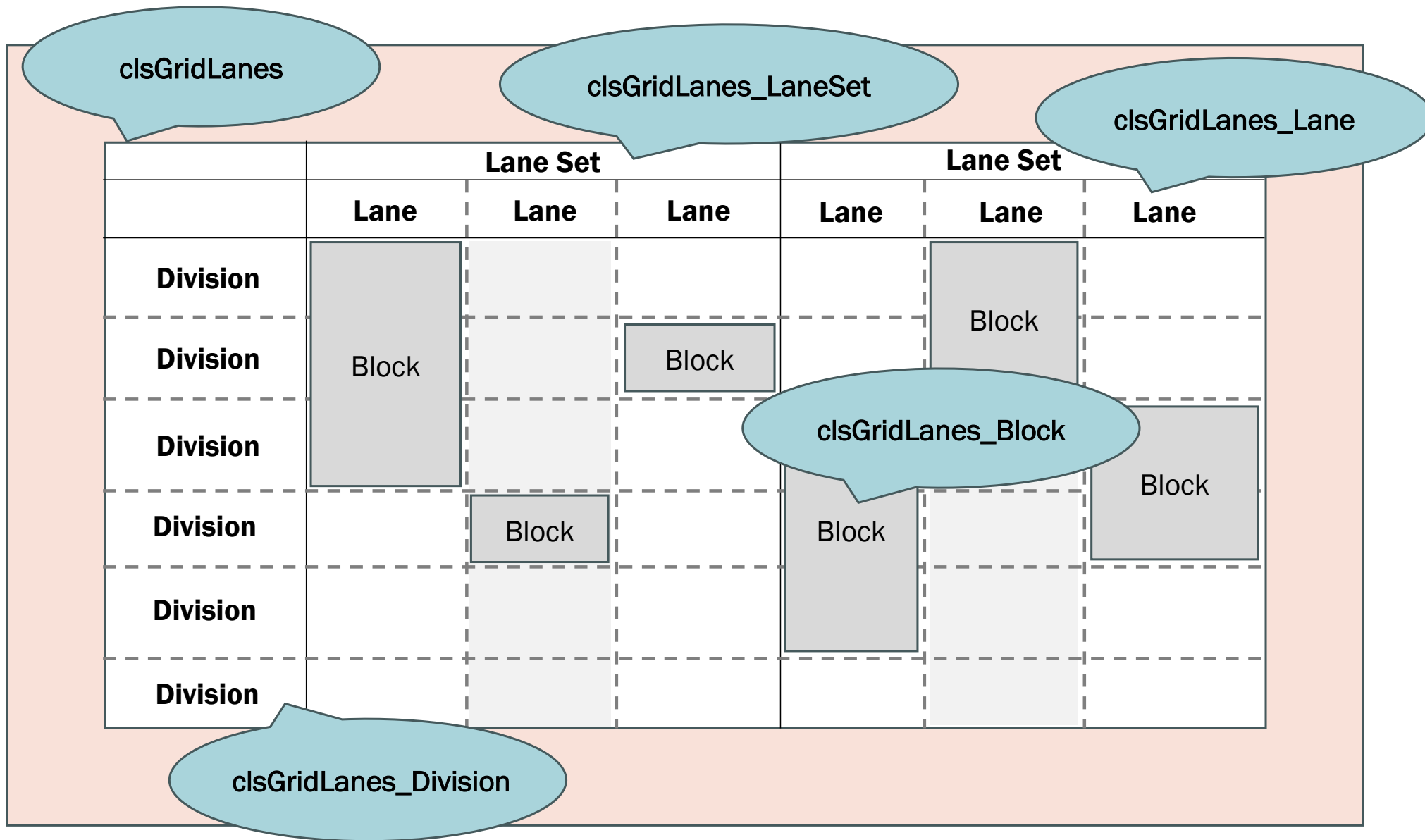
A collection of overlapping geometric shapes in teal, yellow, and olive green, including triangles and diamonds, arranged in a decorative pattern on the right side of the slide.

Design Overview

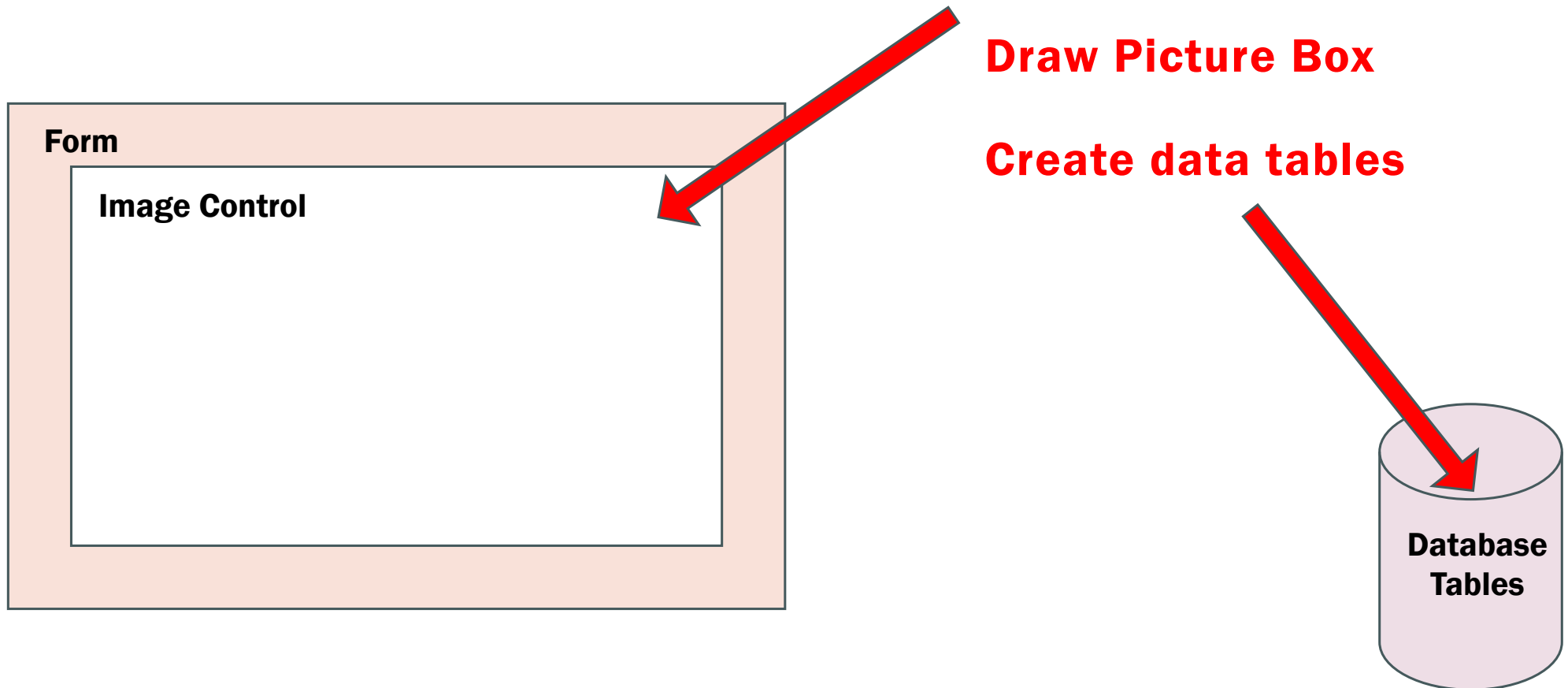
The Classes

Focus in on VBA Dragging Block code

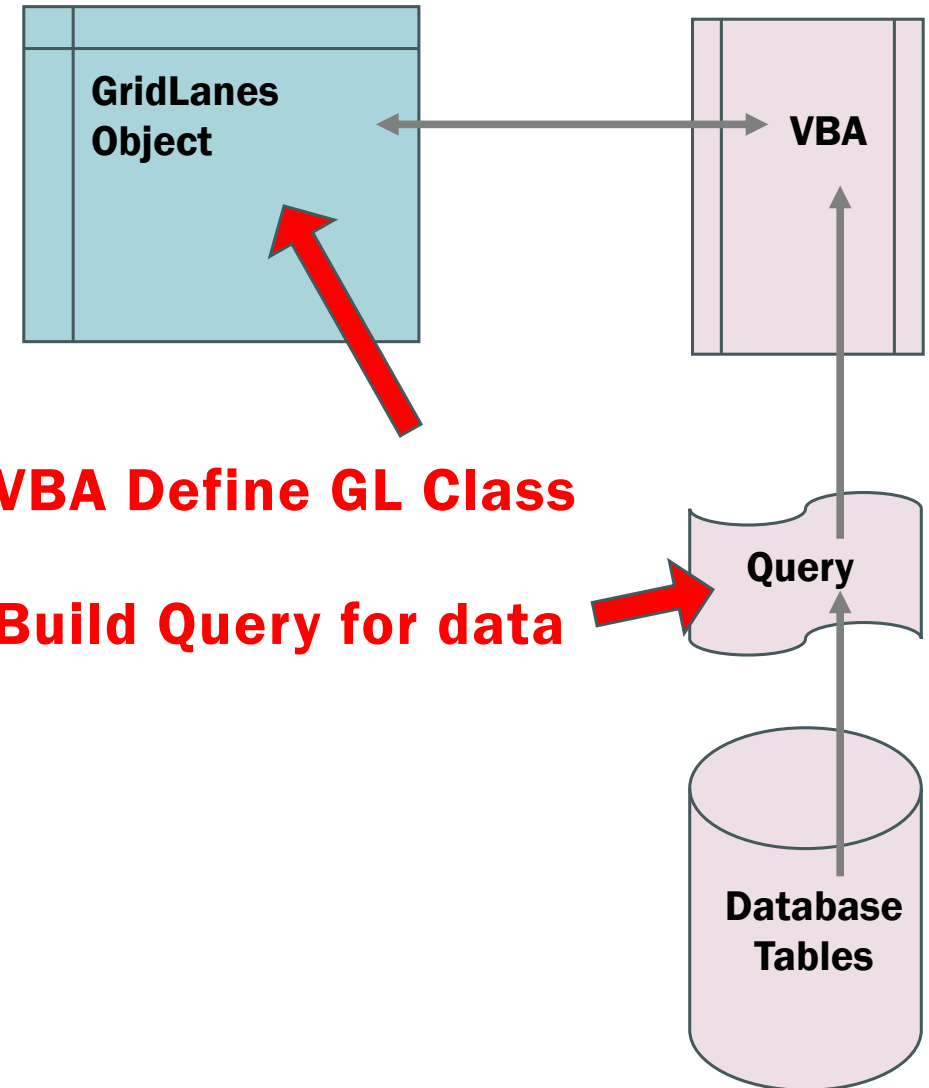
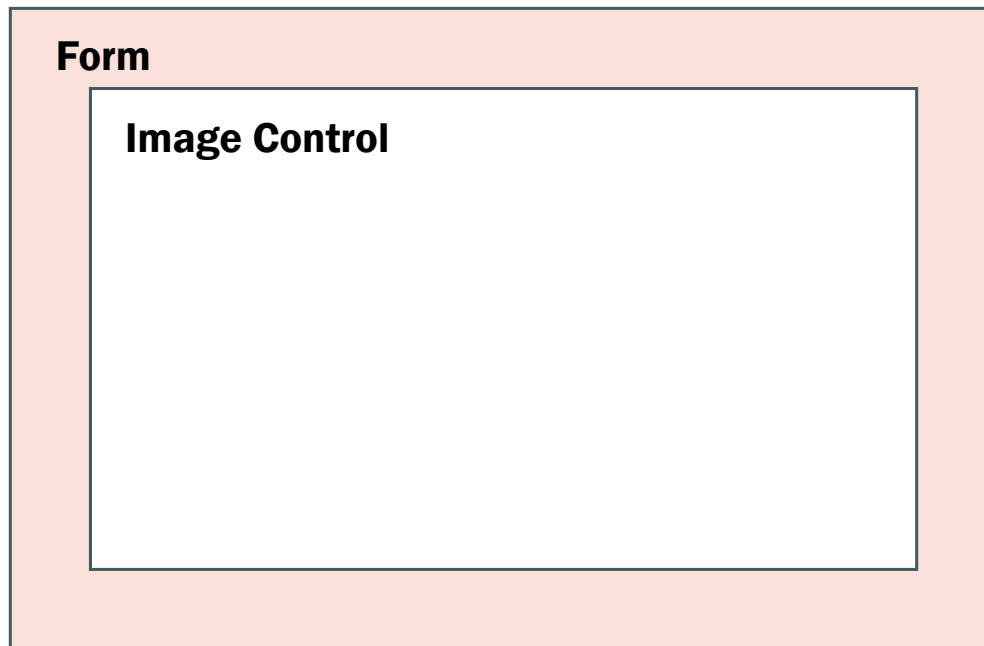
A collection of overlapping geometric shapes in teal, yellow, and olive green, including triangles and diamonds, arranged in a decorative pattern on the bottom left of the slide.



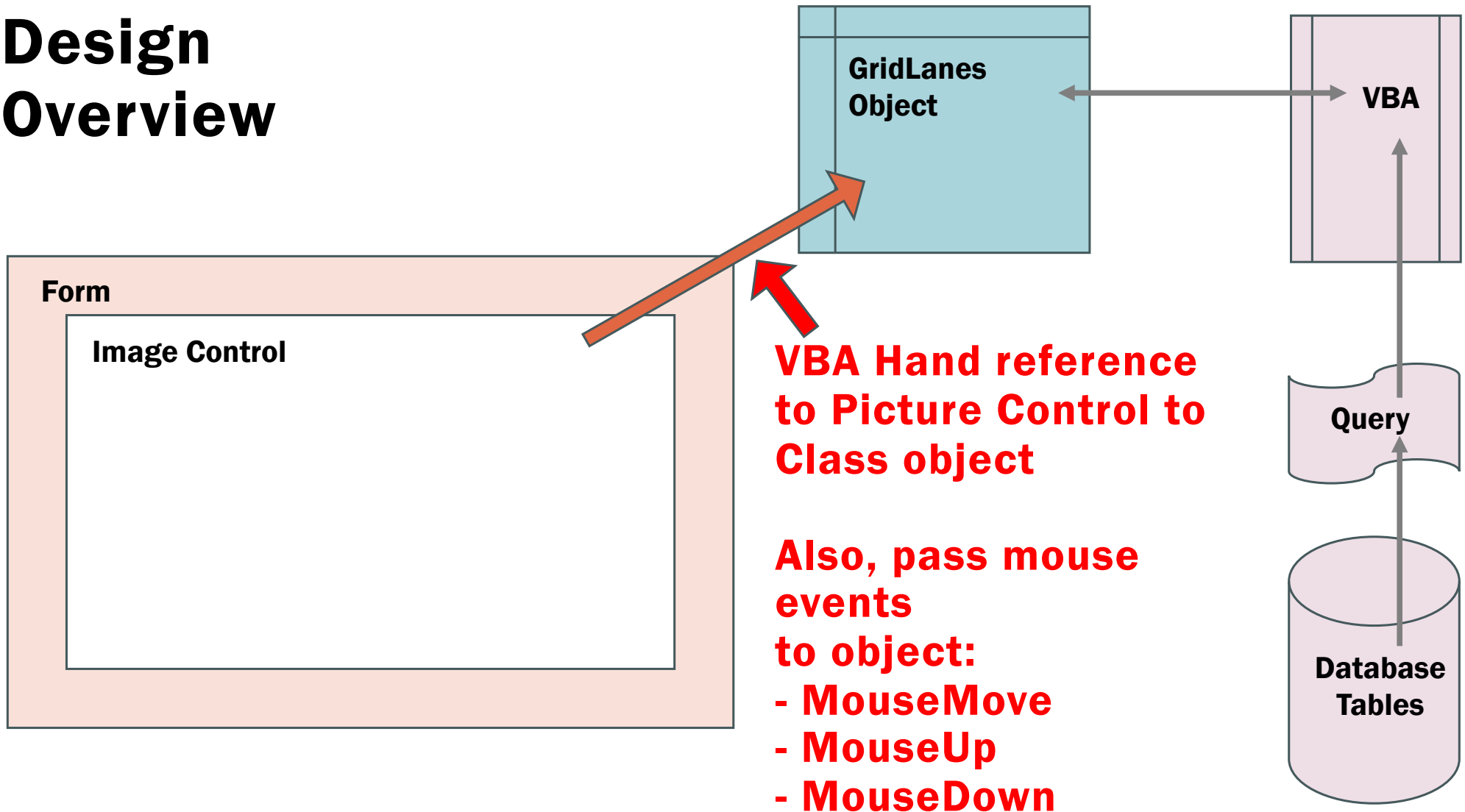
Design Overview



Design Overview



Design Overview



Example Initialisation Code

From an earlier version of the code base

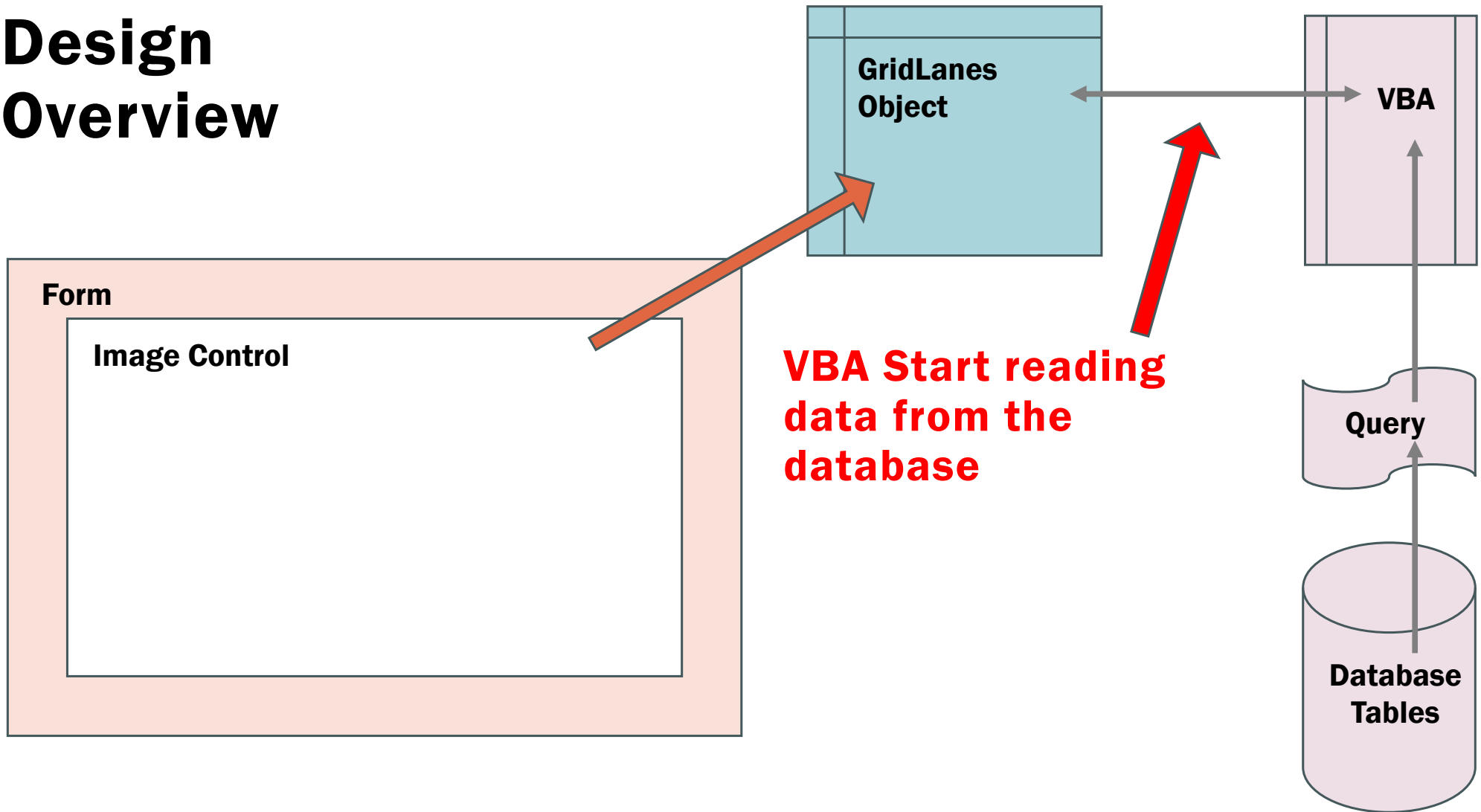
```
Sub subPrepareGridLaneObject()  
    '  
    ' Set up GridLanes object reference  
    '  
    If cvsGridLayout Is Nothing Then  
        Set cvsGridLayout = New clsGridLanes  
  
        '  
        ' Blank out the Picture Control  
        ' and remove any entry stored in the Tag Property  
        '  
        Me.imgGrid.Picture = ""  
        Me.imgGrid.Tag = ""  
  
        '  
        ' Tell GridLanes which Picture control to use  
        '  
        cvsGridLayout.ImageControl = Me.imgGrid  
  
        '  
        ' Also give a reference to the Form the Picture control is on  
        '  
        cvsGridLayout.ImageForm = Me  
  
        '  
        ' Set up some Properties for GridLanes  
        '  
        cvsGridLayout.DrawGridLines = True  
        cvsGridLayout.DrawLaneBackgrounds = True  
        cvsGridLayout.DrawLaneSetHeaders = False  
  
        cvsGridLayout.LaneOrientationIsRow = False  
        cvsGridLayout.LaneHeaderPixels = 60  
        cvsGridLayout.DivisionHeaderPixels = 100  
  
        cvsGridLayout.BackColor = RGB(255, 255, 255)  
        cvsGridLayout.ForeColor = RGB(0, 0, 0)  
    End If  
End Sub
```

ExampleMouseDown code

From an earlier version of the code base

```
Private Sub imgGrid_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If IsObject(cvsGridLayout) Then
        If Not cvsGridLayout Is Nothing Then
            cvsGridLayout.MouseDown Button, Shift, X, Y
        End If
    End If
End Sub
```

Design Overview



Example Load Bookings Code

From an earlier version of the code base

```
bytSuccess = True

Set rcs = CurrentDb.OpenRecordset("qryLoadBookings")
Do While Not rcs.EOF

    bytSuccess = cvsGridLayout.AddBlock( _
        CStr(rcs("BID")), _
        0, _
        rcs("EID"), _
        rcs("RouteOrder"), _
        rcs("Customer"), _
        rcs("Postcode"), _
        "", _
        "", _
        1)

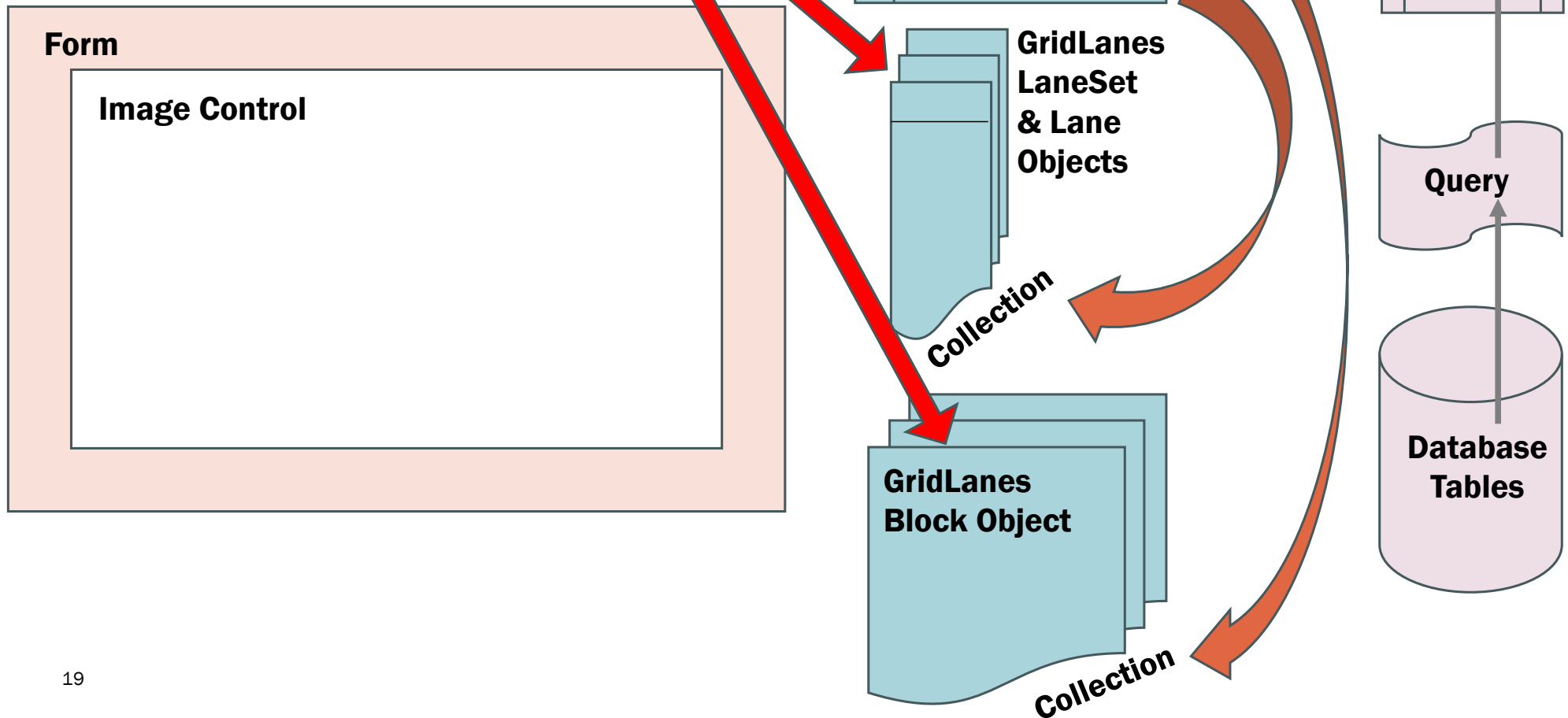
    If bytSuccess <> 255 Then
        Exit Do
    End If

    rcs.MoveNext
Loop

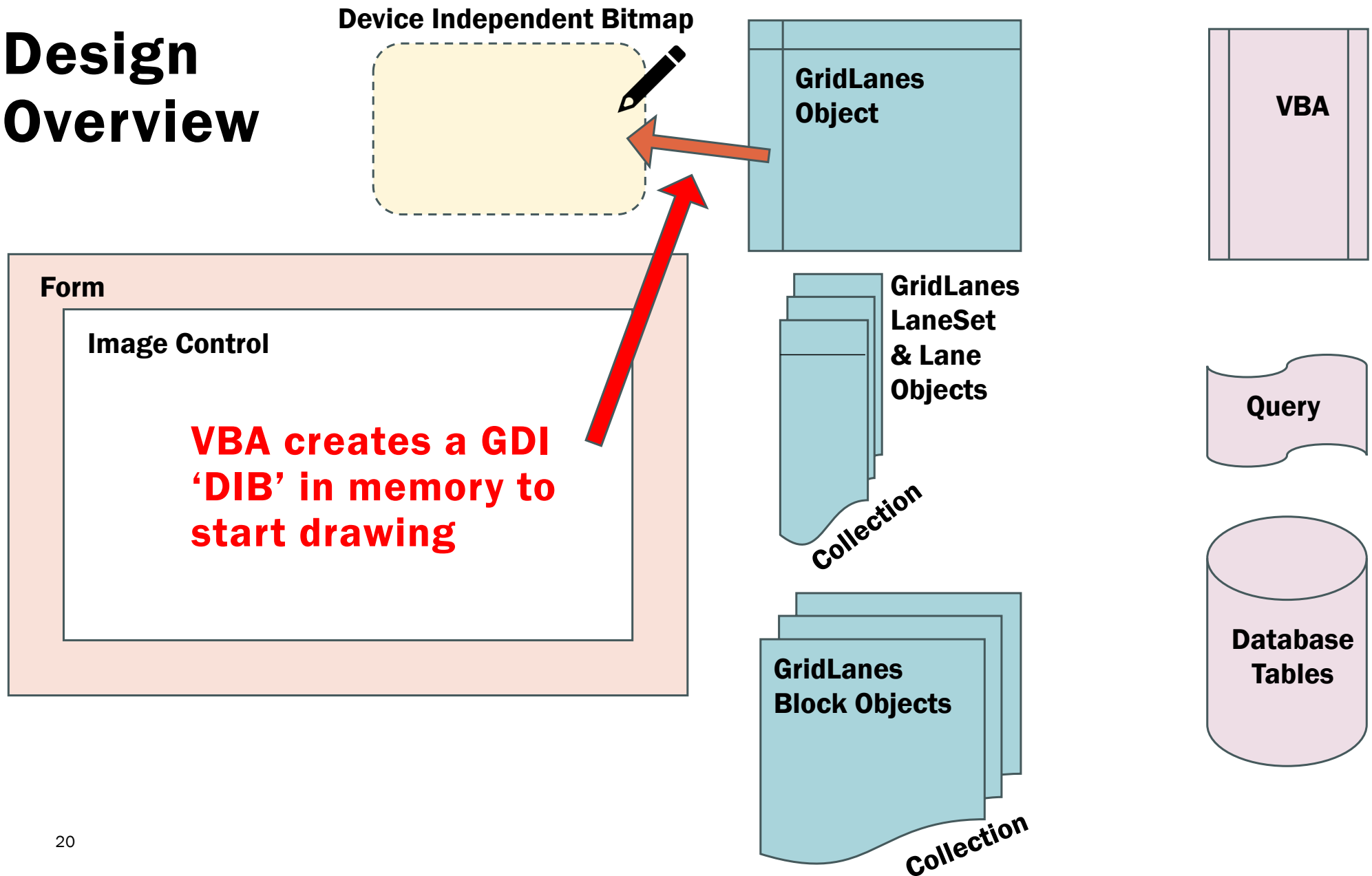
rcs.Close
Set rcs = Nothing
```

Design Overview

VBA Create Lane and Block objects in Collections



Design Overview



Code to start making the DIB, part 1

From an earlier version of the code base

```
Private mhDC As LongPtr
Private mBmi As typBITMAPINFO
Private Const BS_SOLID = 0
Private Declare PtrSafe Function apiFillRect Lib "User32" Alias "FillRect" _
    (ByVal hdc As LongPtr, lpRect As typRECT, ByVal hBrush As LongPtr) As Long
Private Declare PtrSafe Function CreateCompatibleDC Lib "gdi32" _
    (ByVal hdc As LongPtr) As LongPtr
Private Declare PtrSafe Function CreateDIBSection Lib "gdi32" _
    (ByVal hdc As LongPtr, pBitmapInfo As typBITMAPINFO, _
    ByVal un As Long, ByRef lpVoid As LongPtr, ByVal handle As LongPtr, _
    ByVal dw As Long) As LongPtr
Private Declare PtrSafe Function CreateBrushIndirect Lib "gdi32" _
    (lpLogBrush As typLOGBRUSH) As LongPtr
```

Code to start making the DIB, part 2

```
Private Sub CreateDIB()  
    Dim hNewBrush As LongPtr  
    Dim hSystemBrush As LongPtr  
    Dim rc As typRECT  
    Dim lngColor As Long  
    Dim rgbRed As Long, rgbGreen As Long, rgbBlue As Long  
    Dim lb As typLOGBRUSH  
  
    ' mintImageWidthPx, mintImageHeightPx - hold the size of the Image control  
  
    'Create a Device Context  
    mhDC = CreateCompatibleDC(0&)  
  
    ' Create a 24 Bit Bitmap header  
    With mBmi.bmiHeader  
        .biSize = LenB(mBmi.bmiHeader)  
        .biWidth = mintImageWidthPx  
        .biHeight = mintImageHeightPx  
        .biPlanes = 1  
        .biBitCount = 24  
        .biCompression = 0  
        .biSizeImage = mintImageWidthPx * mintImageHeightPx * 3  
    End With  
  
    ' Create the DIB  
    hDIB = CreateDIBSection(mhDC, mBmi, DIB_RGB_COLORS, mlPtr, 0, 0)
```

Code to start making the DIB, part 3

```
' Create a Brush with the background color stored in mlngBackColor
hSystemBrush = 0
hNewBrush = 0

lb.lbColor = mlngBackColor
lb.lbStyle = BS_SOLID

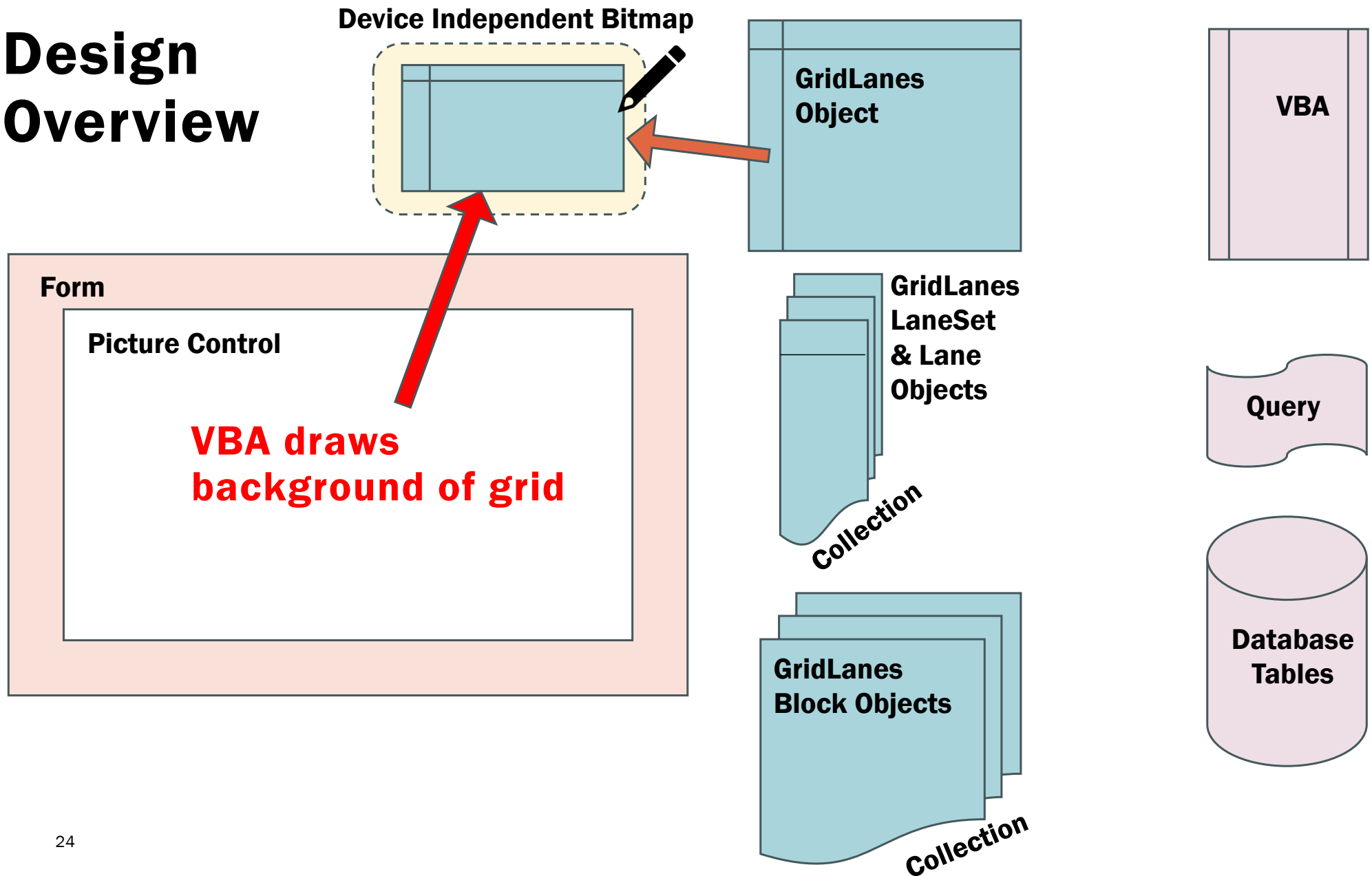
hNewBrush = CreateBrushIndirect(lb)

' Get the Rectangle dimensions from our DIBSection
rc.left = 0
rc.top = 0
rc.right = mintImageWidthPx
rc.bottom = mintImageHeightPx

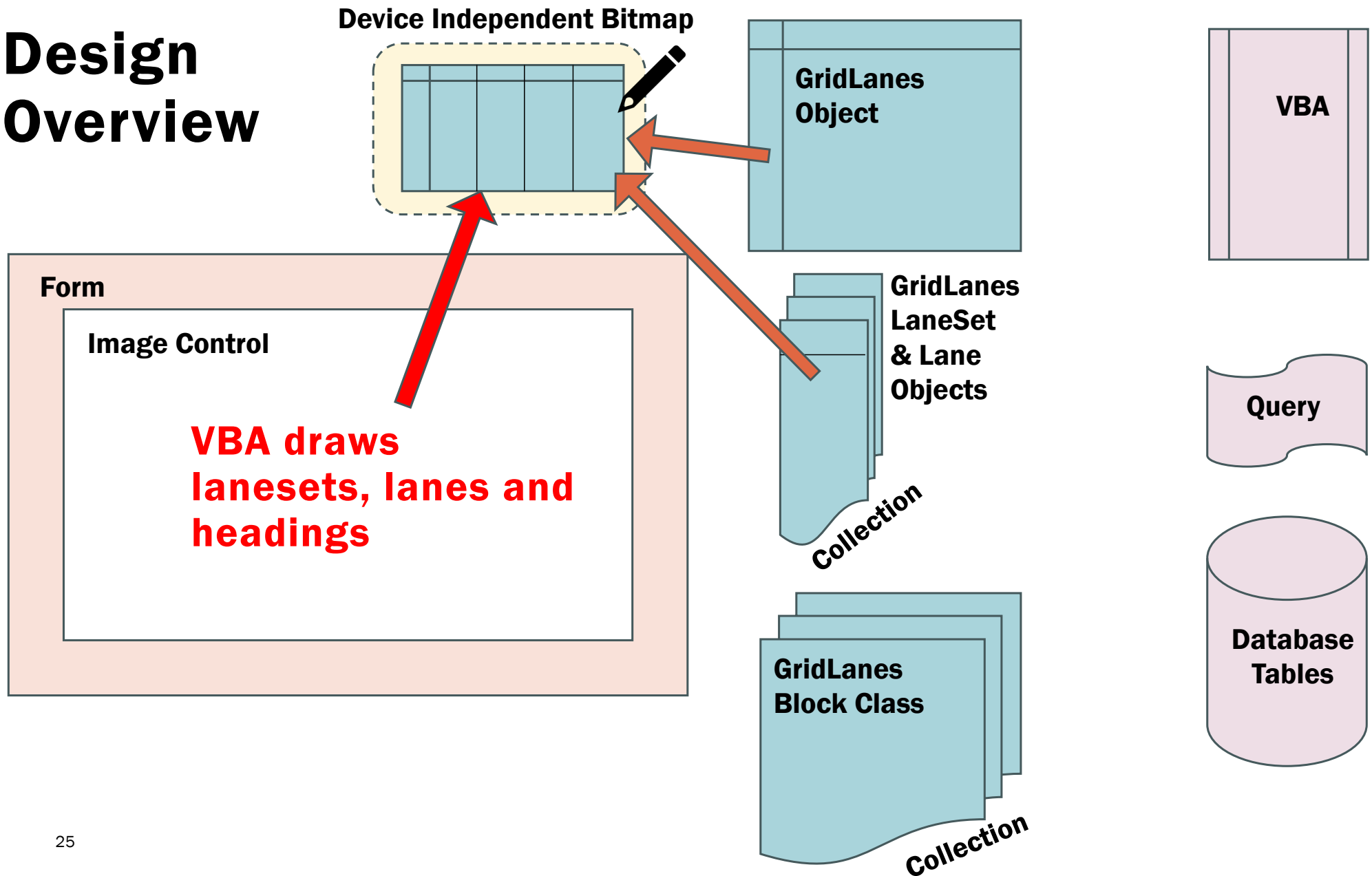
If hNewBrush <> 0 Then
    mlngRet = apiFillRect(mhDC, rc, hNewBrush)
Else
    mlngRet = apiFillRect(mhDC, rc, hSystemBrush)
End If

' Don't delete System Brush only brush created with CreateSolidBrush
If hNewBrush <> 0 Then
    Call DeleteObject(hNewBrush)
End If
End Sub
```

Design Overview



Design Overview



Code to draw rectangle

From an earlier version of the code base

```
Private Sub DrawRectangle(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal X2 As Integer, ByVal Y2 As Integer, _
    Optional ByVal BackColourRGB As Long = &H7F7F7F, Optional ByVal ForegroundRGB = 0, Optional ByVal BorderThickness As Long = 1)
|
    Dim hNewBrush As LongPtr
    Dim hOldBrush As LongPtr
    Dim lngTmpColor As Long

    Dim hNewPen As LongPtr
    Dim hOldPen As LongPtr

    If BackColourRGB > 0 Then
        hNewBrush = apiCreateSolidBrush(BackColourRGB)
    ElseIf BackColourRGB = -1 Then
        ' NULL_BRUSH
        hNewBrush = GetStockObject(5)
    Else
        ' Use FillColor Prop
        hNewBrush = apiCreateSolidBrush(0)
    End If

    ' Select new brush onto our DC
    hOldBrush = SelectObject(mhDC, hNewBrush)

    If BorderThickness > 0 Then
        hNewPen = apiCreatePen(PS_SOLID, BorderThickness, ForegroundRGB)
    Else
        hNewPen = apiCreatePen(PS_NULL, 0, ForegroundRGB)
    End If
    hOldPen = SelectObject(mhDC, hNewPen)

    Call Rectangle(mhDC, X1, Y1, X2, Y2)

    Call SelectObject(mhDC, hOldBrush)
    Call DeleteObject(hNewBrush)

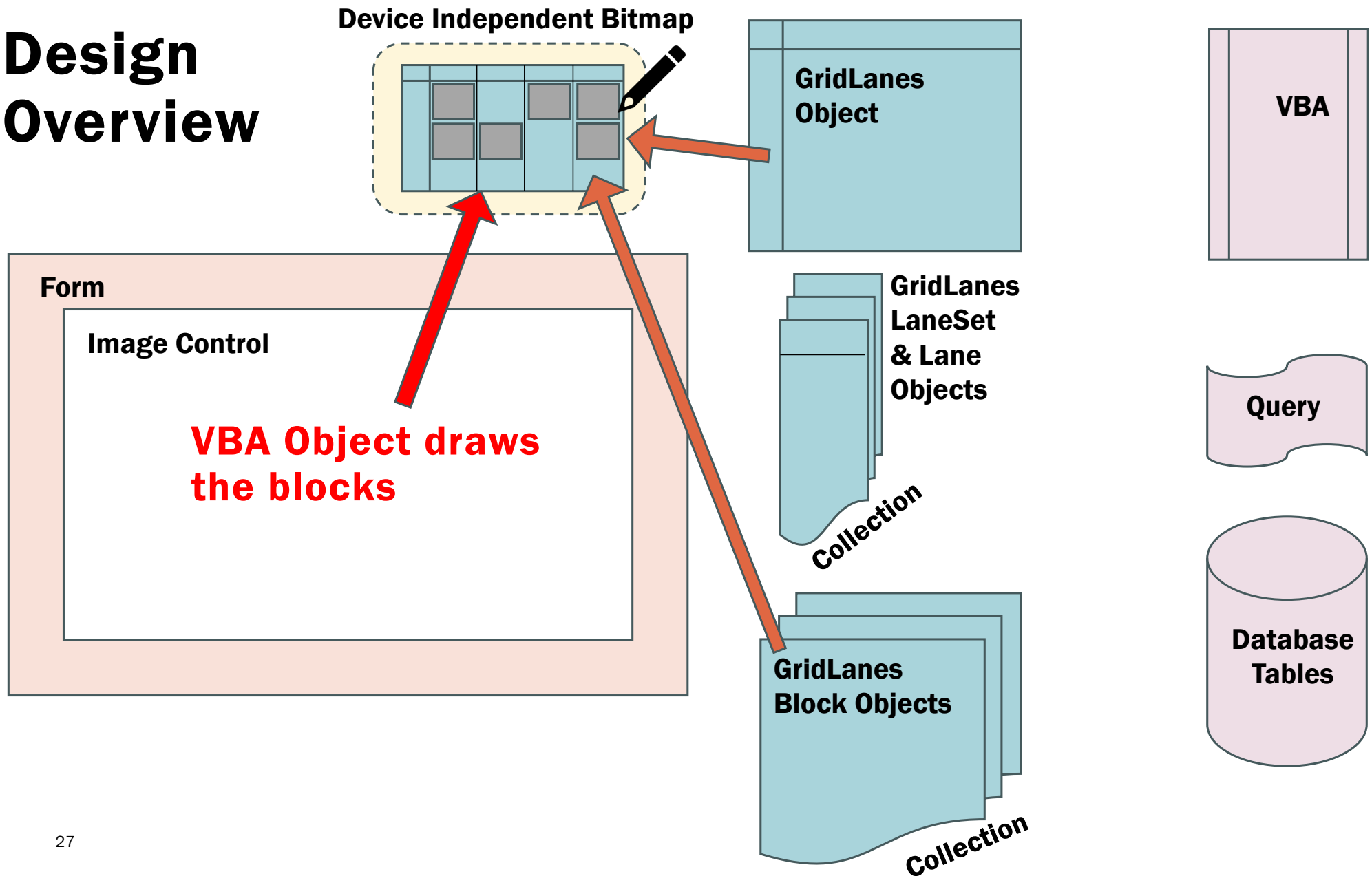
    ' Cleanup
    Call SelectObject(mhDC, hOldPen)
    Call DeleteObject(hNewPen)
End Sub

Private Declare PtrSafe Function SelectObject Lib "gdi32" _
    (ByVal hdc As LongPtr, ByVal hObject As LongPtr) As LongPtr

Private Declare PtrSafe Function Rectangle Lib "gdi32" _
    (ByVal hdc As LongPtr, ByVal X1 As Long, ByVal Y1 As Long, _
    ByVal X2 As Long, ByVal Y2 As Long) As Long

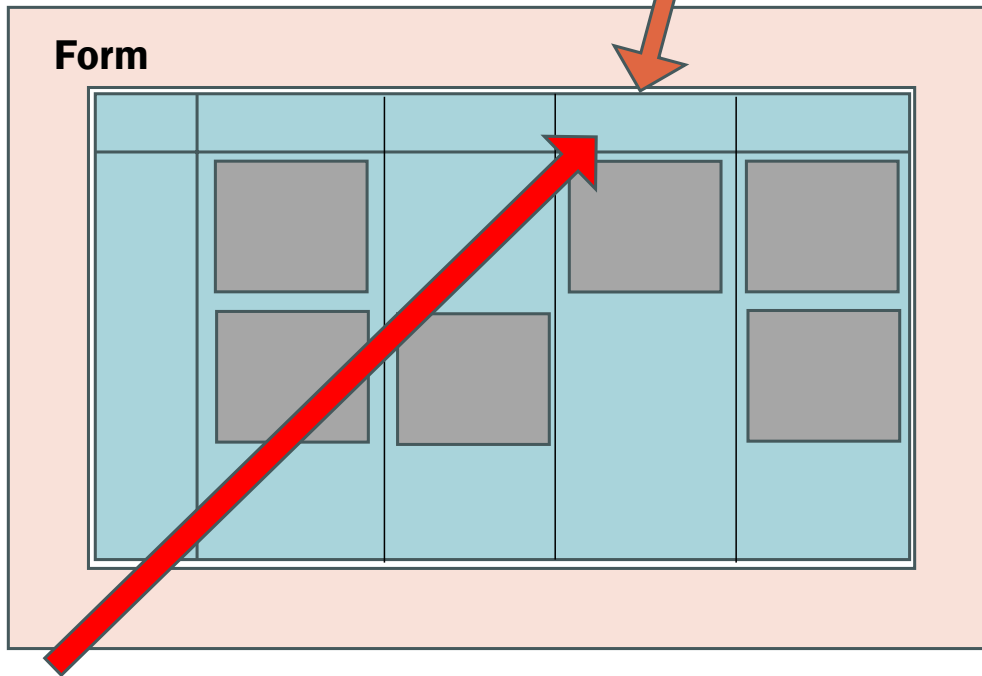
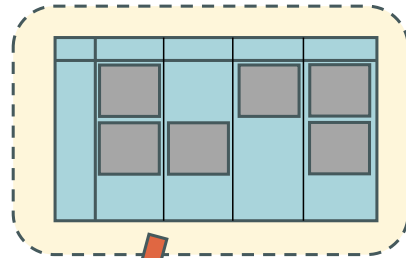
Private Declare PtrSafe Function DeleteObject Lib "gdi32" _
    (ByVal hObject As LongPtr) As Long
```

Design Overview

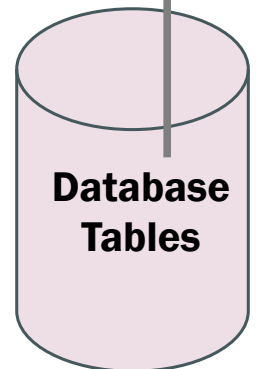
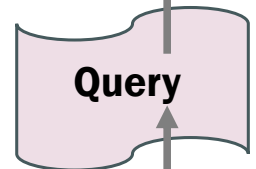
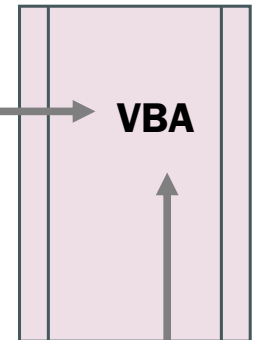
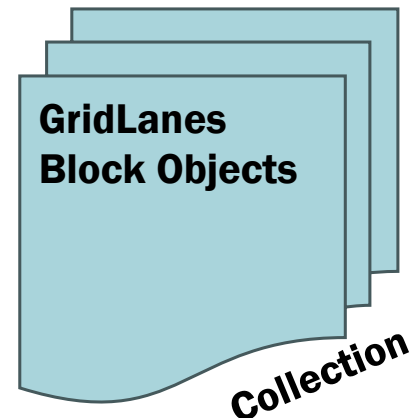
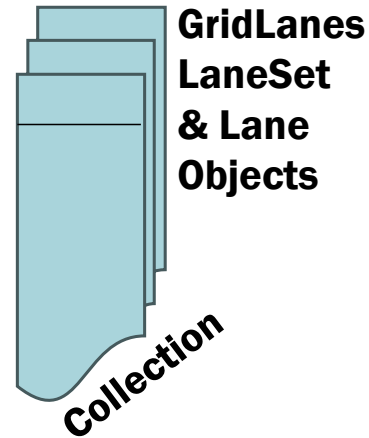
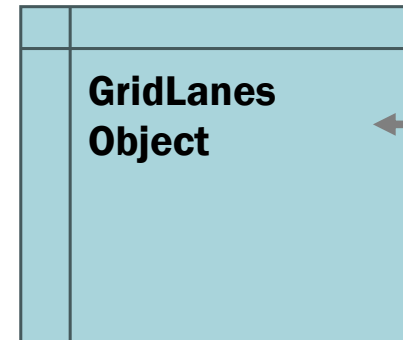


Design Overview

Device Independent Bitmap



VBA DIB data copied to Access form's Image Control .PictureData property



Code to copy DIB to Imagecontrol

From an earlier version of the code base

```
Private Function DIBtoPictureData(Optional DIBnum As Long = 0) As Boolean
    Dim ds As typDIBSECTION

    Dim varTemp() As Byte

    If DIBnum = 0 Then
        ' Fill in our typDIBSECTION structure
        mlngRet = apiGetObject(hDIB, LenB(ds), ds)
    Else
        ' Fill in our typDIBSECTION structure for our Backup DIB
        mlngRet = apiGetObject(mhDIb2, LenB(ds), ds)
    End If

    Dim przepis As Boolean
    przepis = False

    If mlngMemoryPictureTableCounter <> ds.dsBmih.biSizeImage + 40 Then
        ' Allow 40 Bytes for the DIBHeader
        przepis = True
        ReDim mbytMemoryPictureTable(ds.dsBmih.biSizeImage + 40)
    End If

    mlngMemoryPictureTableCounter = ds.dsBmih.biSizeImage + 40

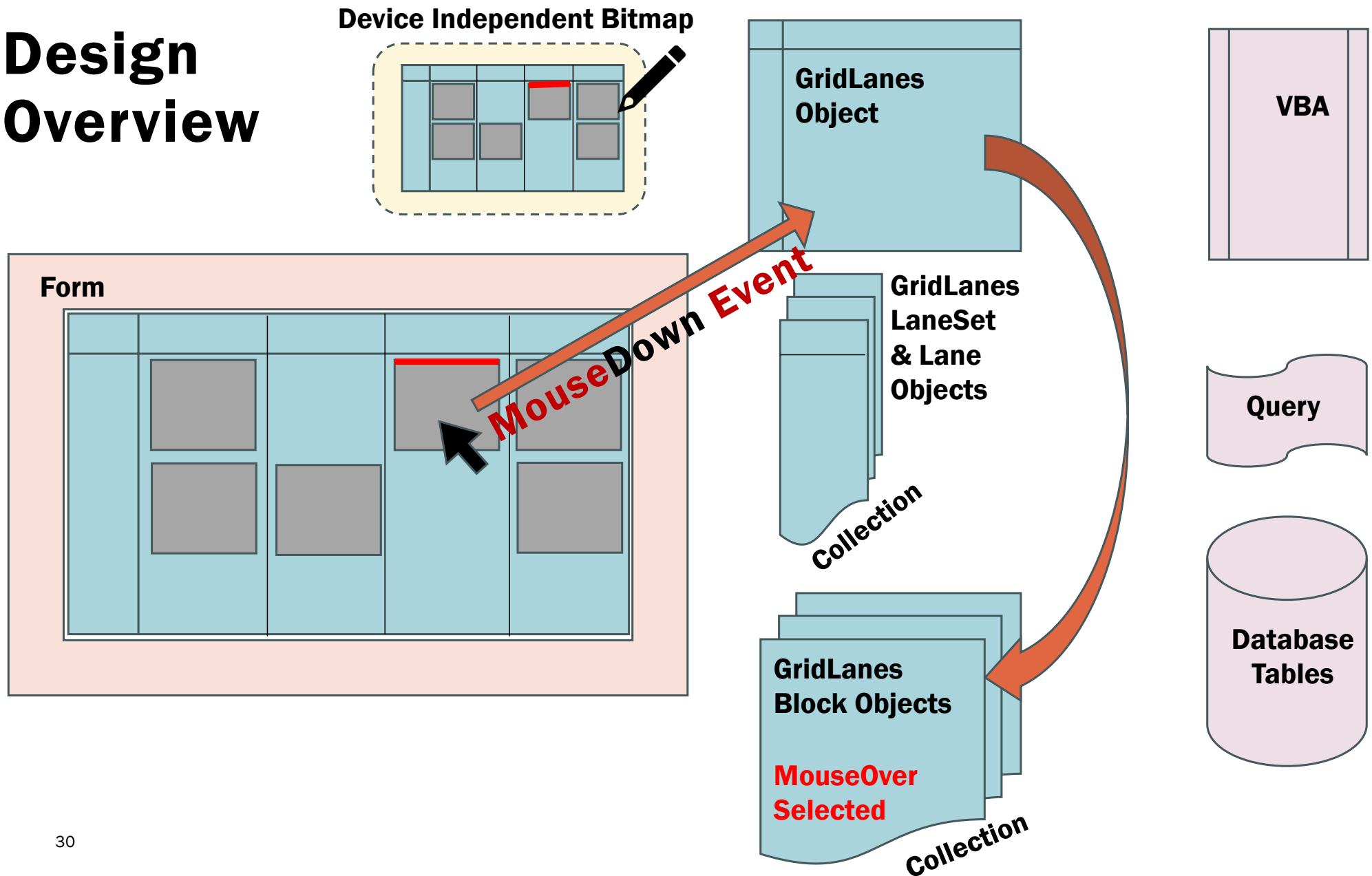
    If DIBnum = 0 Then
        apiCopyMemory mbytMemoryPictureTable(40), ByVal mlPtr, ds.dsBmih.biSizeImage
    Else
        apiCopyMemory mbytMemoryPictureTable(40), ByVal mlPtr2, ds.dsBmih.biSizeImage
    End If

    apiCopyMemory mbytMemoryPictureTable(0), ds.dsBmih, 40

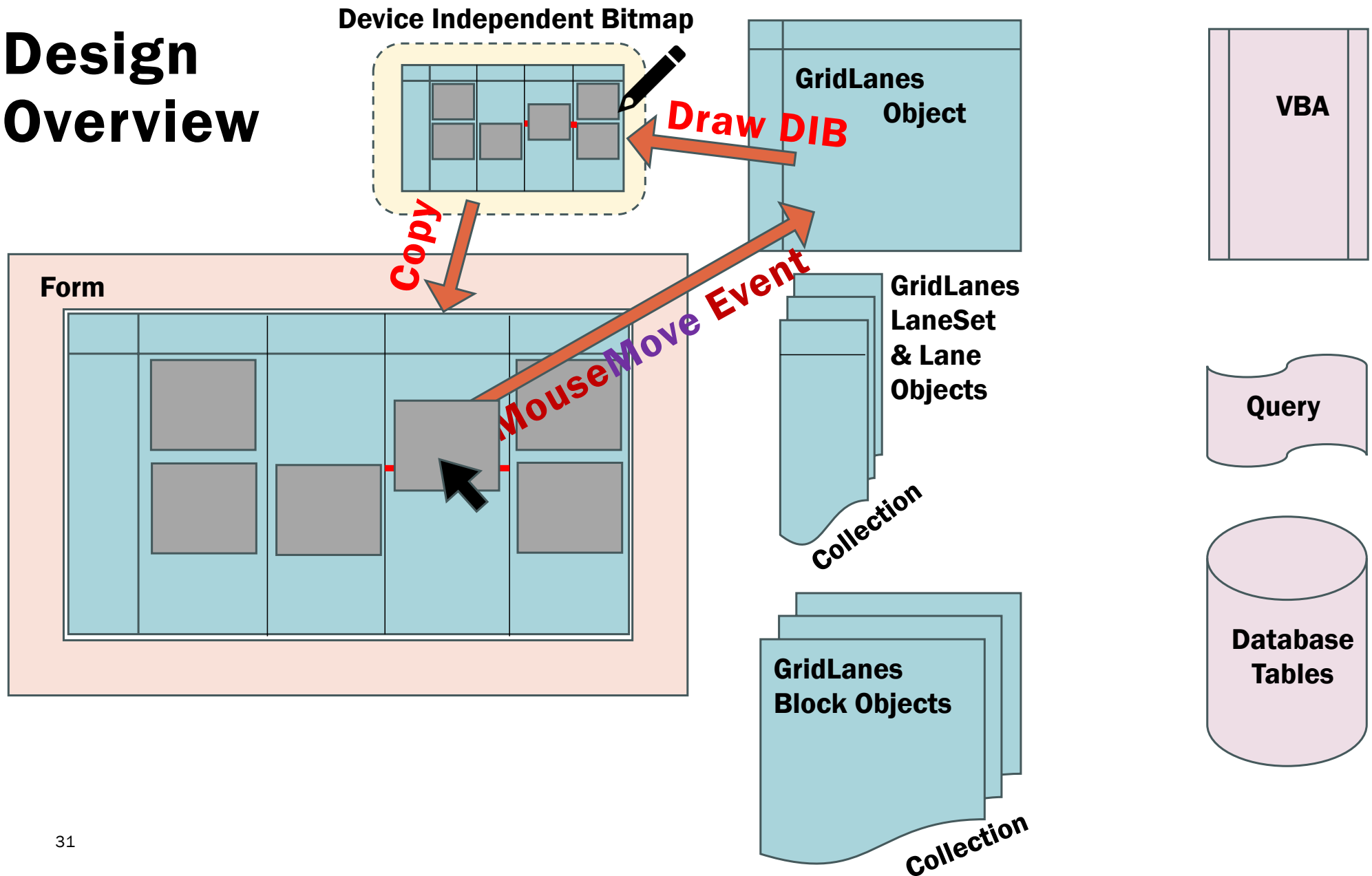
    mImageControl.PictureData = mbytMemoryPictureTable()

End Function
```

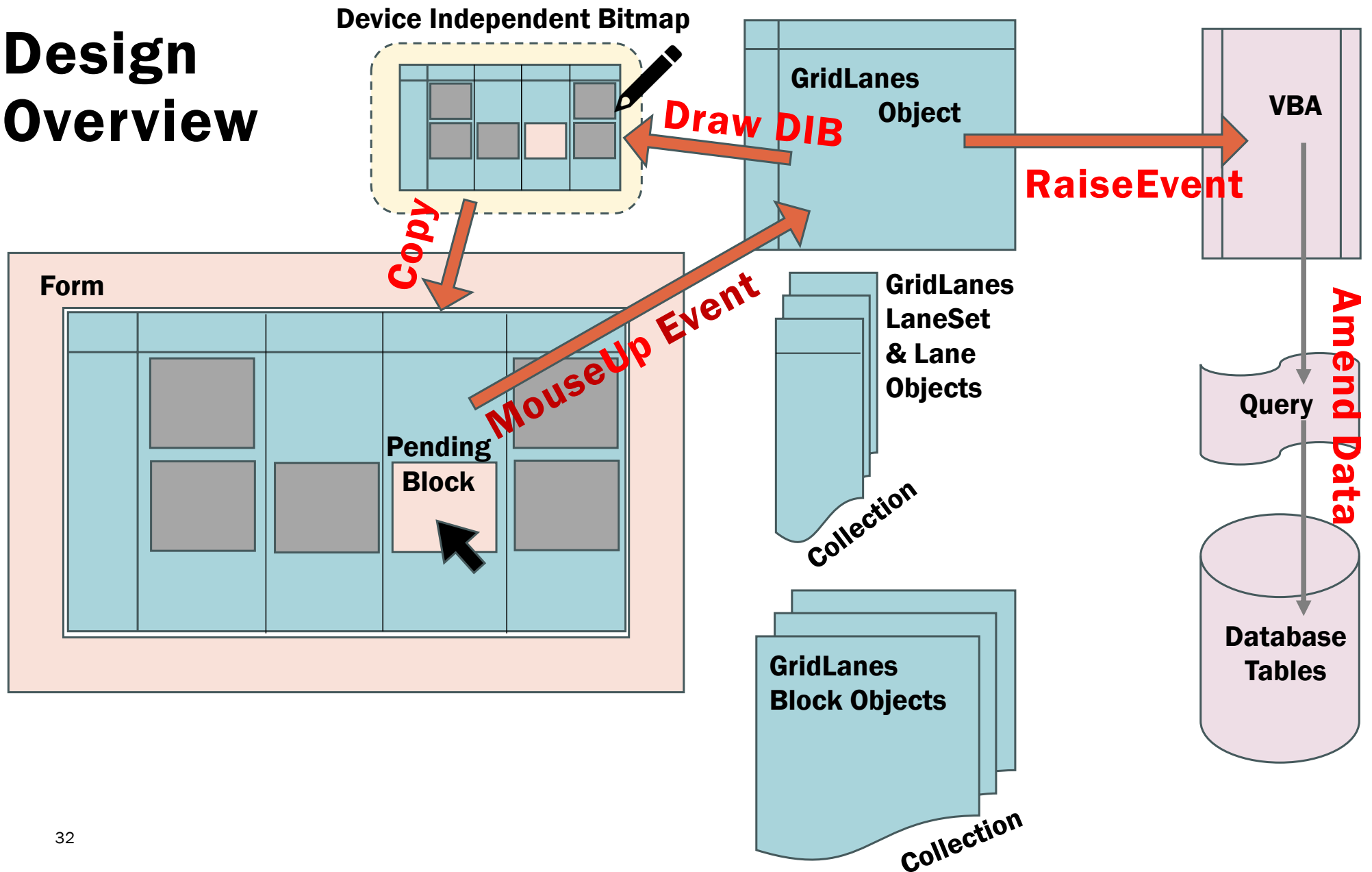
Design Overview



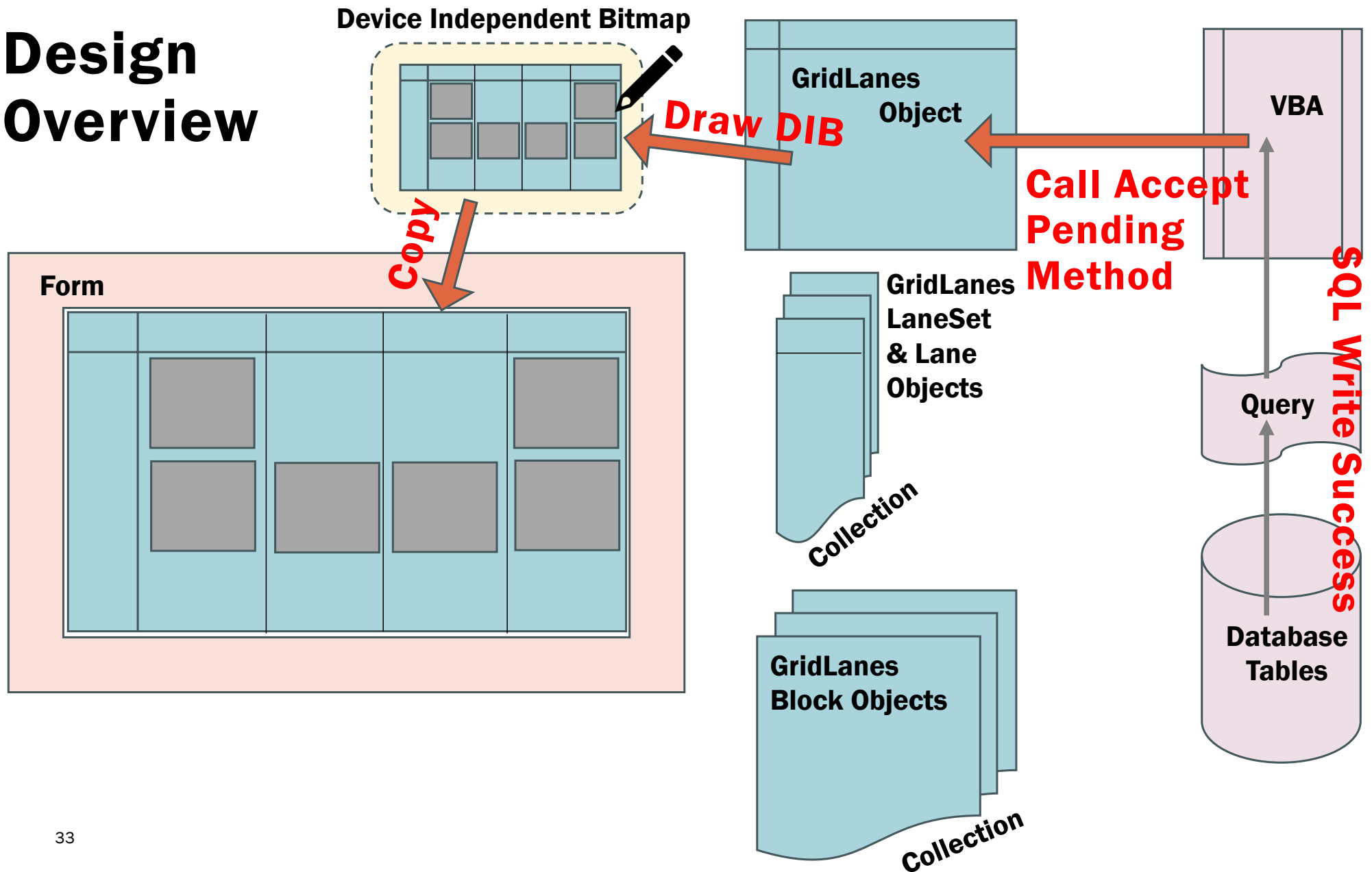
Design Overview

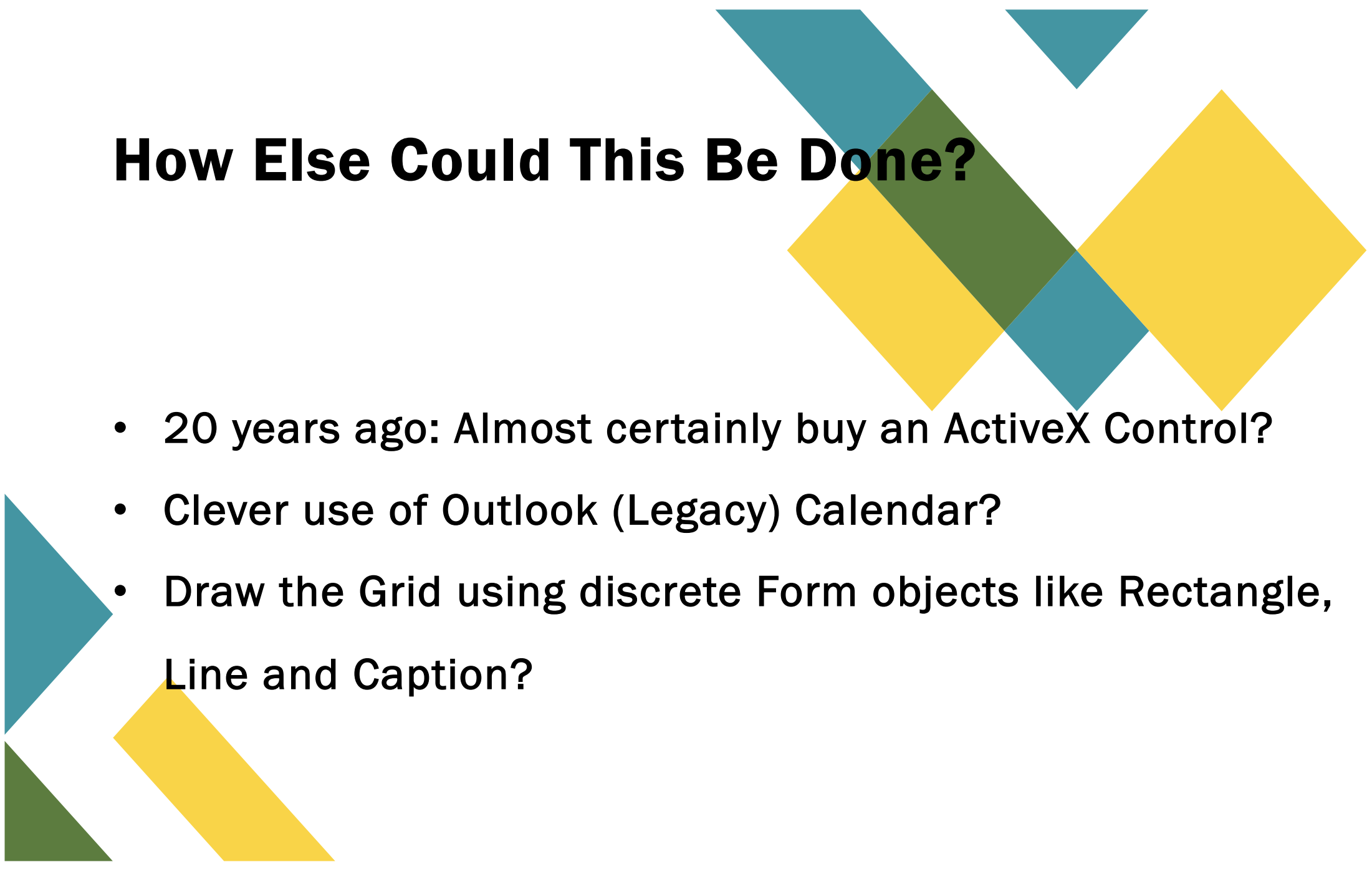


Design Overview



Design Overview





How Else Could This Be Done?

- 20 years ago: Almost certainly buy an ActiveX Control?
- Clever use of Outlook (Legacy) Calendar?
- Draw the Grid using discrete Form objects like Rectangle, Line and Caption?

Future Improvements

- Dragging Flicker – *some improvement just today!*
- Make the control live ‘bindable’ to data
 - **RecordSource** and **ControlSource** type properties
- **Memory Leaks** – *had an out of memory error after running and breaking the code many times in one session (some months ago and not seen since)*

Future Improvements

- **Cascading Styles Class/Objects**
 - Set style for the whole grid
 - Override by setting for lane set
 - Override for setting for a lane
 - Override by setting for a block
- **Properties to control behaviour**
 - Eg. ReadOnly and NoDrag
 - Again these can be set at the Grid level downwards
 - Multi cell occupancy – although LaneSets could resolve this already
- **Multi Select and move objects in a group**



**Now, what is
so clever
about all this?**

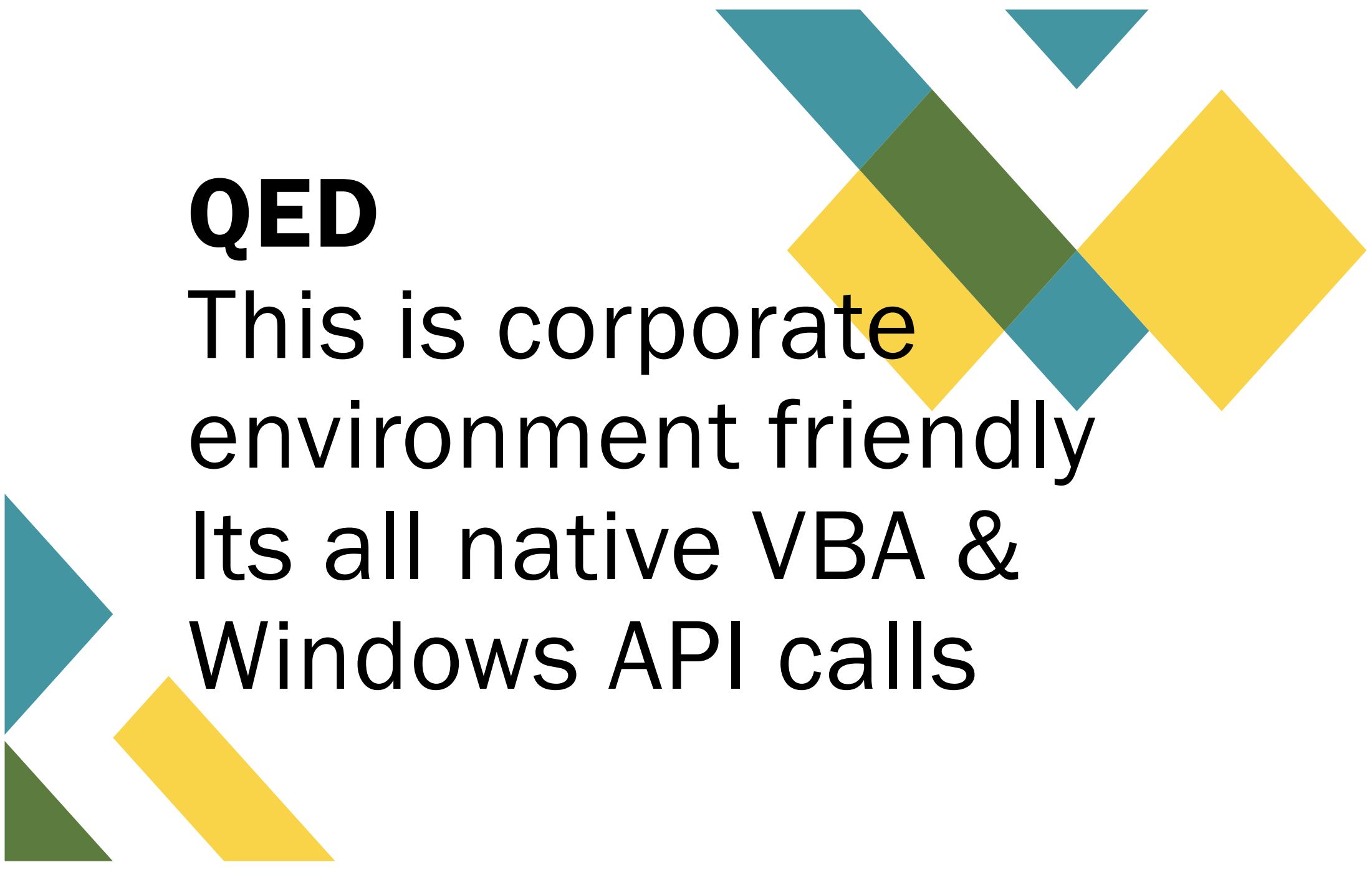


After all....

GDI

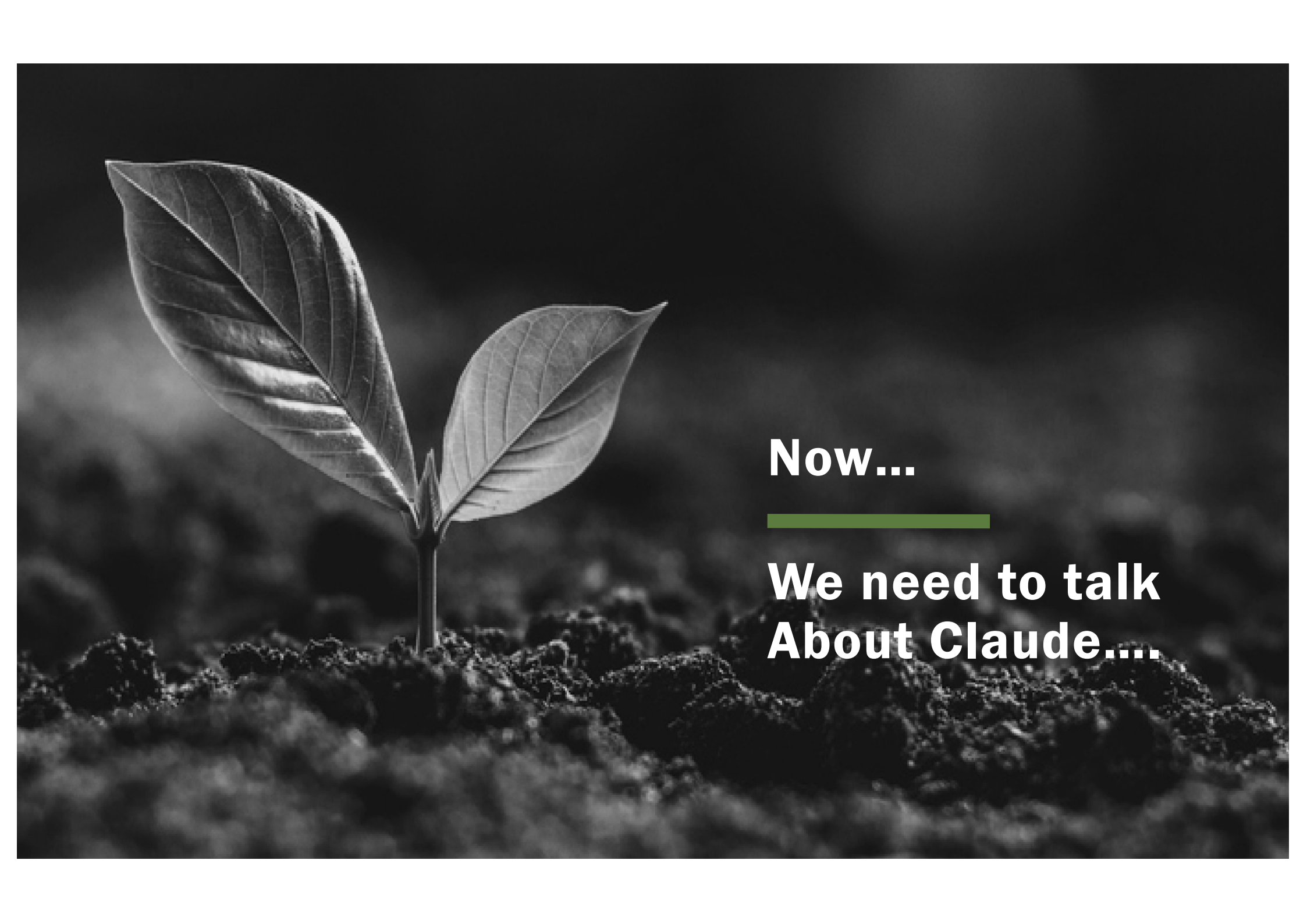
***It's as old as
the hills...***





QED

This is corporate
environment friendly
Its all native VBA &
Windows API calls



Now...

**We need to talk
About Claude....**

Developers

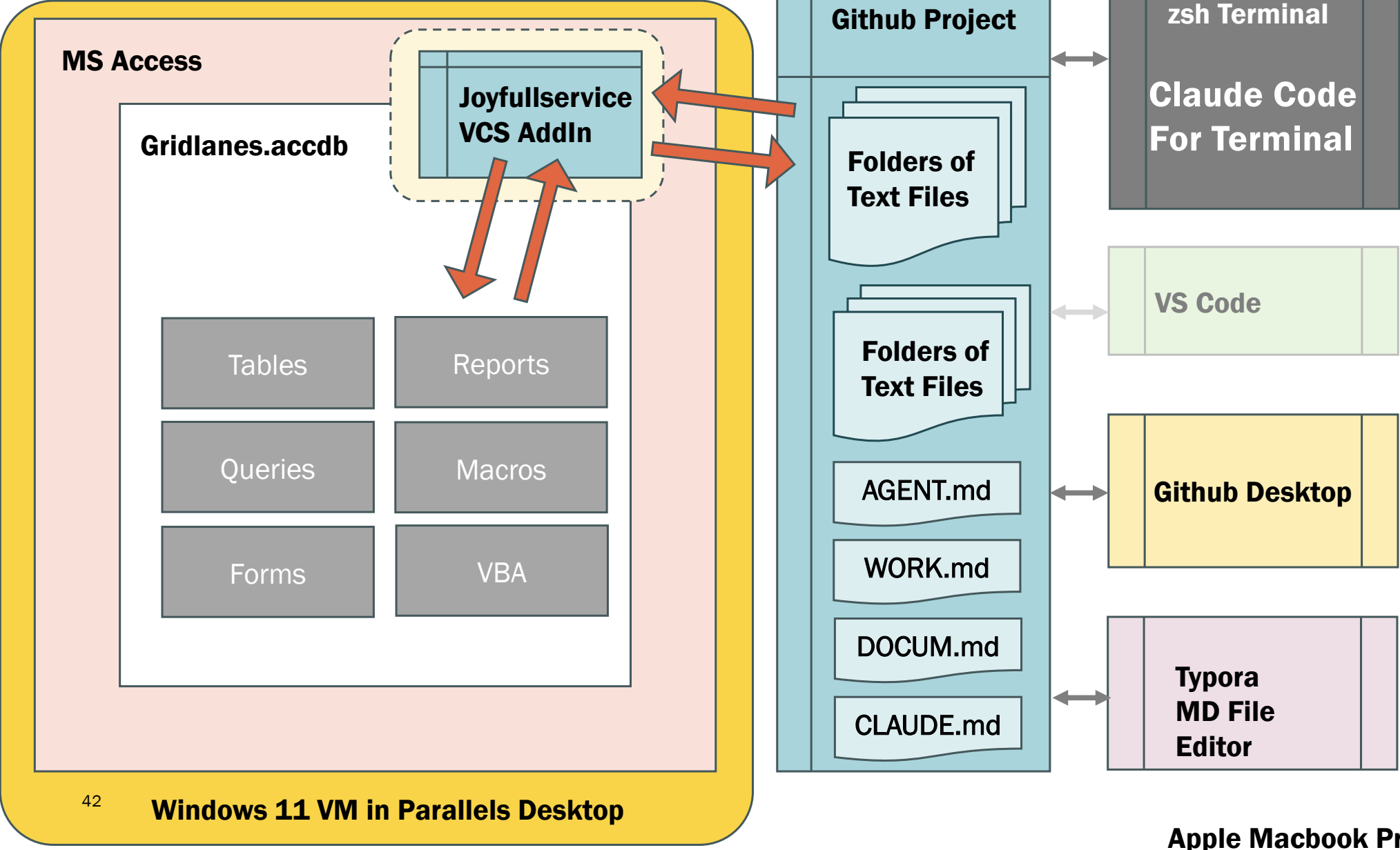
Aleksander

Tim Finch

Claude....



My Current Claude Workflow



**What has
using Claude
been like?**



Experience using Claude

- A bit like a wild stallion horse
- Use of MD files to control what Claude does and limit scope
- Plan, Question then Code. Iterate slowly over features.
- VBA knowledge quite good....
....but writing .form files for Application.LoadFromText ?

```

3   Begin Form
37   Begin
186   Begin Edge
187   TabBorderStyle =1
188   Width =4536
189   Height =2835
190   BackThemeColorIndex =1
191   BorderThemeColorIndex =1
192   BorderShade =65.0
193   GridlineThemeColorIndex =1
194   GridlineShade =65.0
195   End
196   Begin FormHeader
197   Height =793
198   Name ="FormHeader"
199   AlternateBackThemeColorIndex =1
200   AlternateBackShade =95.0
201   BackThemeColorIndex =2
202   BackTint =75.0
203   Begin
204   Begin CommandButton
205   OverlapFlags =85
206   Left =10994
207   Top =170
208   Width =1485
209   Height =498
210   TabIndex =2
211   Name ="cmdRun"
212   Caption ="Run Test"
213   OnClick ="[Event Procedure]"
214

```

Application.SaveAsText

.form files



Experience using Claude

- No longer intimate with the VBA Code. Claude writes it.
 - I am 'Project Director', working with fairly knowledgeable mid-level developer. Who just loves 24/7 working....
...and has slight amnesia
 - Enabled rapid moving forward on various features
 - Produces formal test cases, and reporting workflow, to test
 - User (Programmers) Documentation as a by product
- 

- 13. [Clearing and Refreshing](#)
- 14. [Properties Reference — clsGridLanes](#)
- 15. [Properties Reference — clsGridLanes_Lane](#) (#15-properties-reference--clsgridlanes_lane)
- 16. [Properties Reference — clsGridLanes_Block](#)
- 17. [Known Limitations and Planned Features](#)

1. What GridLanes Does

GridLanes draws a two-dimensional grid of named **lanes** (columns or rows) subdivided into **divisions** (slots). Inside the grid, **blocks** are rectangular labels that the user can drag and drop from one division to another. When a block is dropped, GridLanes fires an event so your code can update the database.

Typical uses:

- An engineer scheduling board — lanes are engineers, divisions are time slots, blocks are customer visits.
- A resource planner — lanes are resources, divisions are dates, blocks are tasks.
- A Gantt-like view — lanes are workstreams, divisions are dates, blocks span multiple days.

The grid is drawn with GDI into a Windows DIB (device-independent bitmap) and then pushed into an Access Image control. Your form passes its Image control and its own reference to GridLanes; GridLanes handles everything else.

2. Concepts and Terminology

Term	Meaning
Grid	The whole control. Managed by the <code>clsGridLanes</code> object.
Lane	One column (or row). Represents a named resource or category.
LaneSet	A named group of lanes. A default lane set is created automatically: call <code>AddLaneSet</code> only when you need

GridLanes Test Results

Results and feedback from running frmGridTest. Each section covers one test case.

Format: PASS/FAIL lines from Debug.Print, then VIEW observations, then any behaviour changes or bugs identified.

Once a TC has been marked as passed please remove it from this document to keep it small.

TC01–TC12 — Confirmed working.

TC14 — Confirmed working (all 9 PASS lines).

TC15 — Confirmed working (3-event sequence correct: P2 displacement, P1 displacement, P3 primary).

TC16 — Confirmed working (3-event sequence correct: Q3, Q2 displaced, Q1 primary).

TC17 — Confirmed working (block snapped back; no displacement events fired).

TC18A — Confirmed working (ANCHOR→D2 notification; FAR stays D4; DRAG primary with 1 displaced).

TC18B — Confirmed working (FAR shifted to div 5; uniform +1 confirmed).

TC20 — Confirmed working (4-event sequence correct: S4, S3, S2 displaced; S1 primary).

TC21 — Confirmed working (FAR→D5 fires first; FIRST→D3 second; DRAG primary; gap no effect on order).

TC22 — Confirmed working (single BlocksHaveChanged per drop; old-position pattern; Lane ref and OldDiv correct across successive cross-lane drags; no visual artefacts).

TC13: Row-orientation rotated LaneSet header (VIEW) — STANDING BUG

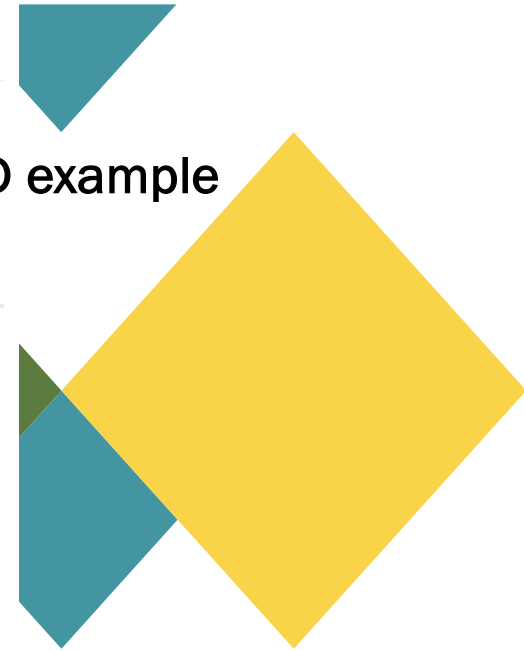
Standing visual bug — deferred, not priority. Lanes and labels render incorrectly in row-orientation. Do not fix in current phase cycle; record for a future phase.

TC19: DIB Transfer Method (F1) — API method observations

VBA method: Working correctly.

API method (screen DC): Y-coordinate fix applied (Section(1) for Form Header height). Grid now renders at correct position. Observed ~1px offset vs VBA method (expected — DPI rounding in twips→pixels→screen coordinates conversion). Content erased on WM_PAINT as expected (not persistent). Higher Z-order windows overdrawn when mouse moves over the Image control area.

Test Results MD example



TC17: Displacement rejection — no room — awaiting explicit confirmation

Debug output showed setup message and no BlockHasMoved events (= correct rejection behaviour). Yellow block should have snapped back visually. Please confirm PASS or describe what you saw.



RESULTS:

Yes it jumped back to where it came from.

=====

TC17: Displacement rejection — no room

DO: 4 divisions. Lane 1 has a block at div 4 (last slot).

DivisionExtensionMode = glDivisionExtendNone.

EXPECT: Dragging any block to lane 1 div 4 would push the existing block to div 5 which exceeds the 4-division limit.

Drop is REJECTED; dragged block returns to original position.

ACTION: Drag the YELLOW block (lane 2, div 1) onto lane 1, div 4.

Block must snap back — no displacement occurs.

TC17 ready — drag yellow block to lane 1 div 4.

Expected: drop rejected; yellow block snaps back to lane 2 div 1.





Gridlanes

Main Processing MD

Class and Feature Improvement

Phase

Please note the Overview and other segments that were in this document have been moved to GridLanesReference.md - when you need to update any of that file please do so.

In this file we will simple keep this Phase segment - where I am giving you the next phase of work to do, and the questions segment.

Where work requested in this Phase segment means to change the Plan & Workstreams , amend them in GridLanesReference.md and note those changes in this file when reporting back on any Phase of work.

New bugs that need my attention write in this file.

Phase 20

Standing bug — TC13 (B15): Row-orientation rotated LaneSet header text. Still wrong after two fix attempts (Phase 16 and 17). Tim confirmed results are "wildly wrong" — deferred to a future phase. Not priority.

TC18 redesigned — split into TC18A (CloseGapOnDisplacement=True, pack) and TC18B (False, uniform). Both are now self-contained; no manual VBA changes required to run either part. Awaiting test results.

Pending test results: TC17 (awaiting explicit confirmation), TC18A, TC18B, TC20.

Phase 19 — Completed



- Done. Here's the summary:

What changed:

- **TC14, TC15, TC16** removed from GridLaneTestResults.md – confirmed passing
- **TC13** recorded as a standing visual bug (deferred)
- **TC18** split into **TC18A** (pack mode, `CloseGapOnDisplacement=True`) and **TC18B** (uniform mode, `=False`) – both self-contained, no manual VBA changes needed. Dropdown updated to 21 items.
- **TC19** results documented; F1 API experiment conclusions recorded
- **GridlanesWork.md** updated with Phase 20 standing bug and pending tests note
- **Version bumped to 1.14**

Still needed from you:

- Confirm TC17 passed (yellow block snapped back visually)
- Run TC18A, TC18B, TC20 with the new code and paste results

Note: GridlanesWork.md still says "moved to Claude.md" in its header – now that the detailed content is in GridLanesReference.md, that reference is stale. Worth a quick manual edit to the intro when you have a moment.

* Cogitated for 3m 54s

※ **recap:** GridLanes VBA grid control improvement project, currently cleaning up test results after Phase 19. Next action: Tim to confirm TC17 passed and run TC18A, TC18B, and TC20 in frmGridTest. (disable recaps in /config)

> █

Claude Code for Terminal on macOS

Before i do this, what workstreams have we not touched yet? I know we have implemented some out of order from your original recommendation.

- From the CLAUDE.md summary, here's where things stand:

Fully done:

- **A+C** – Bug fixes, Movable rename groundwork (Phase 10)
- **E** – Collections migration, LaneSet class, object-reference API (Phase 12)
- **F1** – Dual DIB transfer (`UseAPITransfer`, screen DC fix) (Phase 18/19)

Partially done (started out of recommended order):

- **H** – Displacement on drop: the algorithm, `AllowDisplace`, displaced block events with reverse order are all in. What's still missing: insertion bar (visual vbRed bar showing where a block will land), and full `AcceptPendingChange` integration.
- **I** – `GL_TRISTATE` properties added (`CloseGapOnDisplacement`, `AllowDisplace`, `CloseGapOnDragOut`, `DivisionExtensionMode`). What's still missing: the full cascade (`Resolved*` via parent-ref injection, `Locked` downward-propagating, all objects frozen during pending, method call silently cancels pending).

Not started:

- **G+J** – Style sheets (`clsGridLanes_Style` class), Division class, text-prefix removal, `AcceptPendingChange` with `MissingDivisionIDsWarning`
- **B** – Edge resize: `AllowResize`, `BlockHasResized` event, resize handle drawing
- **D** – Row orientation completeness (TC13 is the standing bug here)
- **F2/F3/F4** – Drag double-buffer and remaining rendering performance
- **K** – Serialisation (lowest priority, last)

The original recommended order was `A+C` → `E` → `G+J` → `I` → `B` → `H` → `D` → `F` → `K`. We've jumped ahead on H and I whilst skipping G+J and B. G+J is the biggest remaining chunk and would unblock the full I cascade. B (resize) is self-contained and could go anytime.

* Worked for 37s

Claude Code for Terminal on macOS



Thank you



Questions?

