

Visual Basic Programmer's  
Journal  
and Microsoft Corporation  
present VBITS 1994



# Optimizing Visual Basic Code

Scott Swanson  
Product Manager  
Applications Programmability  
Microsoft Corporation

[Scottsw@microsoft.com](mailto:Scottsw@microsoft.com)

[vboptimz.zip](#)



---

# How Powerful is Visual Basic?



# Visual Basic is this Powerful!

---



# Optimization Philosophy

---

Understand the real problems.

Finding a good algorithm is better than  
tweaking a bad one.

Consider all the dimensions:

Speed

Size

Maintainability



# Knowing What to Optimize

---

## Walk your code

Where is time being spent?

Where is memory consumed?

## Don't over-optimize

Example: sorting

Example: disk access



# Kinds of Optimization

---

Real speed.

Display speed.

Apparent speed.

Size in memory.

Size of graphics.



# Optimizing Actual Speed

---

Variables are 10 to 20 times faster than properties.

Use Integers and integer math.

Swap tune:

- Put related code in the same module.

- Reduce the number of inter-module calls.

- Keep modules small.



# More Speed Optimizing

---

File I/O: Binary much faster than  
Text/Random.

Use the value of the control.

Strings

Avoid Copying

Temp strings



# Optimizing Data Access

---

**Use Transactions for Bulk Operations:  
BeginTrans & CommitTrans**

**Limit the number of records that you “Visit”**

- Keys are kept in memory

- Snapshots vs. Dynasets

- MoveLast touches every record.

**Attach external databases to Access db's so  
that Table structure is cached.**

- Append a new TableDef to the Database with the  
correct SourceTableName and Connect

**Optimize [ISAM] settings in VB.INI. See  
PERFORM.TXT.**



# Optimizing Display Speed

---

Turn off ClipControls.

Use AutoRedraw appropriately.

Use Image instead of Picture box.

Use Line instead of PSet.

Hide controls when setting many properties to avoid multiple repaints.



# Optimizing Apparent Speed

---

Keep forms hidden but loaded.

Use progress indicators.

Pre-load data you expect to need.

Use timers to work in the background.



# First Impressions

---

Use Show in Form\_Load event.

Simplify your Startup form.

Don't load modules you don't need.



# Keeping It Small

---

Don't use Variants or fixed strings.

Reclaim string and object variables.

Use Dynamic arrays, and reclaim memory when you're done.



# Keeping It Small, continued.

---

Put related code in the same module.

Unload forms.

Remove dead code.

Use string constants instead of literals.



# Cutting Back on Graphics

---

Reclaim memory with LoadPicture() and Cls.

Use Image instead of Picture Box.

Load pictures only as needed, and share pictures and icons at run-time.

Use RLE bitmaps (good) or metafiles (better).

Get rid of icons you don't use.



# Optimizing OLE 2 Operations

---

## Activating applications

Is it already loaded?

Use CreateObject()

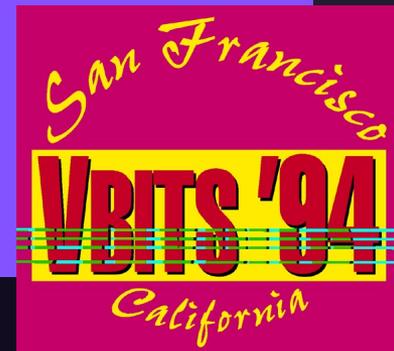
Don't Use DDE or Shell

## In-place Editing

Is application visible?

OLE Automation

Consult the ODK



# When All Else Fails...

---

## Tricks:

“Lurking apps” that never unload.

Multiple apps that act like a single application (using DDE or files).

## Write some DLLs:

Put strings in a DLL and load on demand.

Put graphics in a DLL and load on demand.

Include the most time-critical code.

