Subject: Tools or No Tools ?
Posted by Oz on Sun, 05 Oct 2014 19:49:01 GMT
View Forum Message <> Reply to Message

When it is time to consider writing Apple IIgs video games or a demo, the first question is usually to decide which would be the best environment (Operating system) to use.

There are 3 available options :

- Proprietary OS
- Prodos 8
- GS/OS

Everyone who has watched the FTA Demos (Nucleus, Modulae, XMas, Delta...) / Games (Space Harrier, Bouncin Ferno, Oil Landers...) / Utilities (Photonix, NoiseTracker), the Miami Software (aka F.*.C.K) products (Teenage Queen, Sensei, Show 3200, Hot Cookies, Space Shark, ZZ Copy...) or some of the Mr Z Demo (California Demo, Pom's / Toolbox disk introduction, Crack Intro...) have been amazed by what a basic Apple IIgs could do. For a moment, the slow computer they were facing everyday with Prodos 16 was muted into a computer where everything was fast, in color and in stereo ! Most of these products use a proprietary OS, very light and very fast to be loaded from a 3,5 inches disk.

Most of the game, even the best ones like Rastan, were based on a Prodos 8 system. Few of them, especially the latest ones, were GS/OS compatible (Out Of this World, Wolf 3D, Ultima I) and NOT protected against the copy.

The real question is about GS/OS. Is the choice of GS/OS a constraint or an advantage for Game programmers ? Why so many people were ignoring GS/OS when they where writing their games ? Could we write also Demos under GS/OS ?

It is important to understand that programming a video game under GS/OS won't prevent you to use all the required tricks for fast sprites animation, scrolling & co. Using GS/OS DOES NOT MEAN using Quickdraw to write pixels on the screen. Using GS/OS, it is just about allocating memory and making sure we do not blow up other programs memory spaces. The 'No Tools' and 'No GS/OS' is not the same. You can program low level and use the GS/OS to allocate for you the right memory space before using it. And when it will be time to quit the game, free the allocated memory. There is no constraints to use GS/OS. Only advantages (SynthLab / Tool 219 / Tool 220 musics).

The only reason why mot of the game were using Prodos 8 at the time was linked to the small memory size of the Apple IIgs (in the USA, some of them had only 512 KB). So, in order to keep the maximum memory space for the game itself, the choice of Prodos 8 was done (it is also to have the maximum space on the 800 KB disk, where Prodos 8 was taken less space than Prodos 16 or GS/OS). For anyone having 2 MB or more, GS/OS is much more friendly environment. You can install your games on your hard drive, launch them and go back to the finder when exit. Because the hard drive is faster than the disk, the games loading is faster. Today, with the use of emulators, using a image disk for a game (and booting from the disk) is painful.

The specific OS, like the ones we had at the Apple II time, has been use mostly for demos or games. Because such programs do no require to write something back to disk, the usage of the disk was only done by fast read-block access. This is fast but once again, such program can't benefit of a better Apple IIgs configuration. They are limited to the usage of the 800 KB disk drive and can't be installed on a hard drive (which would be faster, at the end, than a 800 KB drive).

In order to show that programming under GS/OS in not complex, you will find next a skeleton of program. Build it with Merlin 16+ and you will get a S16 executable file. Launch it from GS/OS. Hit a key to quit. The program does nothing spectacular but this could be a base for any game or demo under GS/SOS.

The code itself is self-explanatory, there is nothing tricky. You can notice that the only thing we really need, as game programmer, is an access to the $01/2000 area, which is the graphic page once the Shadowing is active. We simply need to ask GS/OS for this memory area. Because the Apple IIgs is a single process at a time system, once your program is running you are the only one doing something and you have full CPU power. In the case GS/OS still need to interrupt your program (AppleTalk, Sound Interrupt, Graphic Interrupt...), you can protect yourself by using the SEI /CLI opcodes. All code between a SEI and the CLI can't be interrupted, so you can do what you want (we think here about the move of the Stack & Direct Page in Bank $01 for fast graphic routines).

Feel free to ask any explanations about this small code set.

```
*-----   Merlin 16+ Directives

         mx      %00              ; assemble in 16 bit

         rel                    ; Build a relocated S16 file
         dsk     Hello.l          ; Name of your program on disk

         use     4/Locator.Macs     ; Macro Definition Files
         use     4/Mem.Macs
         use     4/Misc.Macs
         use     4/Util.Macs
         use     4/Sound.Macs

*-----   Begin Of Program   ----------

         PHK                      ; Data Bank Register = Program Bank Register
         PLB

         CLC                      ; 16 bit
         XCE
         REP      #$30

         JSR     ToolInit          ; Init Tools + Compact Memory + Ask Shadowing
         JSR     BackupEnv           ; Backup environment (colors...)
```

```
*-----  Your Code Starts Here   ----------

        JSR     WaitForKey        ; Wait until a Key is pressed

*-----  End Of Program   ---------

End       JSR     RestoreEnv        ; Restore environment (colors...)

        JSR     ToolTerm          ; End up Tools
        JMP     Exit              ; Quit to the Launcher

*******************************************************
*******       INIT TOOL SET/ FREE TOOL CODE       *******
*******************************************************

ToolInit  _TLStartUp                ; Start Tools
        PHA
        _MMStartUp                ; Start Memory Manager Tool Set
        PLA
        STA     myID              ; Get current ID
*--
        _MTStartUp                ; Start Miscellaneous Tool Set
*--
        PushLong  #0              ; Allocate Page Direct in Bank 00
        PushLong  #$000100
        PushWord  myID
        PushWord  #$C005              ; Fixed, Page Aligned, Locked, Unpurgeable
        PushLong  #0
        _NewHandle
        PLX                       ; Handle Low  Address
        PLA                       ; Handle High Address (00XX bank)
        XBA
        STA     TI_1+2
TI_1      LDAL    $000000,X         ; Get Low Address  A=??/XXXX
        PHA
*--
        _SoundStartUp             ; Start Sound Tool Set
*--
        PushLong  #0              ; Compact Memory
        PushLong  #$8fffff
        PushWord  myID
        PushWord  #%11000000_00000000
        PushLong  #0
        _NewHandle
        _DisposeHandle
        _CompactMem
*--
        PushLong  #0              ; Ask Shadowing Screen ($8000 bytes from $01/2000)
```

```
         PushLong  #$8000
         PushWord  myID
         PushWord  #%11000000_00000011
         PushLong  #$012000
         _NewHandle
         PLA
         PLA
*--
         RTS

*-------

ToolTerm    _SoundShutDown              ; Stop Tools
         _MTShutDown
         PushWord  myID
         _DisposeAll
         PushWord  myID
         _MMShutDown
         _TLShutDown
         RTS

myID       ds       2                 ; ID of this Program in memory

*--------------------------------------

BackupEnv   SEP      #$30              ; Backup Environment values (color, border...)
         LDAL     $00C022
         STA      BE_C022
         LDAL     $00C029
         STA      BE_C029
         LDAL     $00C034
         STA      BE_C034
         LDAL     $00C035
         STA      BE_C035
         REP      #$30
         RTS

*-----

RestoreEnv  SEP      #$30              ; Restore Environment values (color, border...)
         LDA      BE_C035
         STAL     $00C035
         LDA      BE_C034
         STAL     $00C034
         LDA      BE_C029
         STAL     $00C029
         LDA      BE_C022
         STAL     $00C022
```

```
          REP      #$30
          RTS

BE_C022    HEX      00               ; Background Color
BE_C029    HEX      00               ; Linearization of the Graphic Page
BE_C034    HEX      00               ; Border Color
BE_C035    HEX      00               ; Shadowing

***********************************************************
*******               GS/OS CODE               *******
***********************************************************

GSOS       =       $E100A8

*-------

Exit       JSL      GSOS             ; Quit Program
          dw       $2029
          adrl     gsosQUIT

*-------

gsosQUIT   dw       2                ; pCount
          ds       4                ; pathname
          ds       2                ; flags


***********************************************************
*******          EVENT HANDLER CODE            *******
***********************************************************

WaitForKey  SEP      #$30             ; Wait for a Key Press
WFK_1      LDAL     $00c000
          BPL      WFK_1
          STAL     $00c010
          REP      #$30
          RTS

***********************************************************
```

Subject: Re: Tools or No Tools ?
Posted by mcpderez on Sun, 05 Oct 2014 23:09:41 GMT
View Forum Message <> Reply to Message

I think it was over on the Facebook group (which seems impossible to search) someone was showing The Programmer's Introduction to the Apple IIgs. Then someone said most of what was in there is obsolete. Specifically, Sheppy (IIRC) said that new GS software should be written using

TaskMaster which was not covered by that book.

I think this all ties into what OS to use for a game and I suspect TaskMaster is either a tool or a library or paradigm for structuring an event loop.

So, does that book help much with game programming or is it considered obsolete in this domain too? Is TaskMaster something that should be used if selecting GS/OS for the game? I don't know what TaskMaster is, but it just sounds like something that could add too much overhead and slow things down too much. He didn't really say where TaskMaster is documented, but I would not be surprised if it is in one of the Toolbox Reference books that sadly seem unavailable in PDF.

Any hints or nudges in the right direction will be appreciated.

Subject: Re: Tools or No Tools ?
Posted by Oz on Mon, 06 Oct 2014 19:26:07 GMT
View Forum Message <> Reply to Message

Like most of you, I enjoy FaceBook for its capability to let people present on a daily basis how they interact with the Apple II world (retro bright session, Ebay auctions, pictures, video, misc news) but it is the worst place to search for something which is 1 month old. Because Google do not index FaceBook content, everything disappear very quickly. It is impossible to use such space for References or long term work. A forum like this one is perfect to focus on one specific direction (IIgs programming). What we are saying here will still be valid in 10 years.

Quote:someone was showing The Programmer's Introduction to the Apple IIgs. Then someone said most of what was in there is obsolete. Specifically, Sheppy (IIRC) said that new GS software should be written using TaskMaster which was not covered by that book.

Sheppy is right when he speaks about how TaskMaster has changed the way the Apple IIgs programs have to be written. On the early ages of the Apple IIgs (Prodos 16 was the first 16 bit Os delivered with the IIgs), the Apple IIgs programming was very close to the Macintosh programming. Thanks to its small market, the Apple IIgs system developers had the flexibility to add features that the Mac was missing : The TaskMaster, a new way to deal with Graphic Interface events (mouse click, keyboard key press...). The Apple IIgs programming was easier and more elegant than the Macintosh one.

Everything is true, but only if you consider writing applications using the Apple Graphic User Interface (Apple Menu bar, pull down menu, window with check box, radio button, Text Edit, QuickDraw II...). Everything that make an Apple IIgs program looks the same than another Apple IIgs program. For video games, excepts few of them like Balance of Power, ChessMaster, Full Metal Planete, we don't use the Apple Interface but we take control of the whole screen, remove mouse cursor and Menu Bar and handle all the events ourselves.

QuickDraw II routines are tailor made for Windows and 'serious' applications, not suited for action games where sprites have to be drawn very quickly on the screen. The only events we have to handle when we write a game is the keyboard (Dark Castle), the joystick (Rastan) or the mouse (Zany golf) for controlling the character on the screen. Reading the right memory locations to find out which key has been pressed, what is the direction of the Joystick and where is the mouse is easy and can be done directly in assembler without having to involve the TaskMaster.

We will publish here the low levels routines to read such events. Unlike TaskMaster, in low level assembly language we choose to see if something has occurred (keyboard key, joystick direction, mouse status) instead of being notified that something has happen. If you don't go to read the mouse position, nothing will happen if the user try to click or move the mouse. On the TaskMaster, the events are kept by the system and deliver (in the right order) to the process if required.

As conclusion, we won't start the TaskMaster in arcade video game because we don't need it.

Olivier

Subject: Re: Tools or No Tools ?
Posted by mcpderez on Tue, 07 Oct 2014 07:43:46 GMT
View Forum Message <> Reply to Message

Oz wrote: Like most of you, I enjoy FaceBook for its capability to let people present on a daily basis how they interact with the Apple II world (retro bright session, Ebay auctions, pictures, video, misc news) but it is the worst place to search for something which is 1 month old.

Yes, exactly!

mcpderez wrote:someone was showing The Programmer's Introduction to the Apple IIgs. Then someone said most of what was in there is obsolete. Specifically, Sheppy (IIRC) said that new GS software should be written using TaskMaster which was not covered by that book.

OK, I now understand TaskMaster is the event manager under GS/OS, at least more or less.

Oz wrote:As conclusion, we won't start the TaskMaster in arcade video game because we don't need it.

So, how useful is The Programmer's Introduction to the Apple IIgs for arcade video games? Is there a better starting book (that I don't have to buy) besides the French one already posted?

I think I will have a choice to make, or freedom to play with both using TaskMaster or not. I am not very good at fast arcade video games (I die quickly and get frustrated), so I prefer turn-based strategy games. I doubt anything I write at first will be for public consumption, but mostly to show myself I can make something. Maybe even YaTTT (Yet another Tic-Tac-Toe). :d I hope you won't find this a waste of your time; who knows what it will evolve into and I have to start somewhere.

Finally, I had one more question related to the original post in this thread. Are there some examples of suitable "proprietary OS" with released source? Or are these like asking magicians to tell you their tricks?

Now, I need to get busy setting up an environment to try the sample code you posted.

Thank you Olivier!

Subject: Re: Tools or No Tools ?
Posted by Oz on Wed, 08 Oct 2014 12:48:32 GMT
View Forum Message <> Reply to Message

Mark,

Quote:So, how useful is The Programmer's Introduction to the Apple IIgs for arcade video games?

Not very useful. If your game is running under GS/OS, you have very few to know about the OS. Simply allocating memory, loading files, starting few tools (Misc, Memory, Sound...).
Everything will be explained here in the next coming topics. You can read the book, but 95% is related to GS/OS application using the Apple Graphic User Interface. Not what most of the games are using.

Quote:Is there a better starting book (that I don't have to buy) besides the French one already posted?

The first thing to learn is about the Graphic Page organization. On the Apple IIgs this is very simple (one Graphic Page located in $E1/2000, 320*200, 16 colors / Pixel, 4096 color in the palette). Dagen will probably present the details in one of its videos. If it is not the case, we will do a summary here. These explanations can be found in many Apple IIgs books (look inside the Hardware Reference and the Firmware Reference books).

You can also read some of the Tech Notes related to Graphic & Animation. Stay away from any Tools or QuickDraw II explanations, it is for Windowed applications. Not useful for games.

Quote:Maybe even Tic-Tac-Toe. Very Happy I hope you won't find this a waste of your time; who knows what it will evolve into and I have to start somewhere.

The first step is to be able to understand + assemble + test the samples we provide here. Once the Inputs (Read Mouse / Read Keyboard / Read Joystick / Load Files from disk) and the Outputs (Draw on the Screen, Play Sounds, Write a file on disk) are working, you can start to program your game. A Tic-Tac-Toe is fine because you have all previous elements to set up together.

Quote:Are there some examples of suitable "proprietary OS" with released source? Or are these like asking magicians to tell you their tricks?

I have opened a Topic dedicated to Apple IIgs Games Source Code. I have classified them by OS : Prodos 8, GS/OS, Custom...

Quote:I need to get busy setting up an environment to try the sample code you posted.

I think Dagen will cover soon the development environment, including Merlin 16+. This is typically the kind of thing that is better looking a video than reading explanations here.

Olivier

Subject: Re: Tools or No Tools ?
Posted by Dagen on Fri, 10 Oct 2014 02:01:04 GMT

View Forum Message <> Reply to Message

I was very inspired the FTA when I was young and took the idea of "No Tools" very literally. This was a BAD idea. A friend and I made some cool demos, not quite as good as the French groups, but not bad for two 16-year-olds. We went the route of installing a bootloader and building a custom OS as Olivier describes in his post. When the size of our binary changed, we had to change the number of blocks loaded in the bootloader. We had no toolchain to just copy files since we weren't using a proper OS, so we'd just be copying blocks of data to specific disk locations and updating our loader params. It was very cumbersome. And now, I haven't seen many in years because they require a working 3.5" drive (not hard drive installable) and generally had other ROM 1 specific issues and didn't use the memory manager so I'm sure we stomped on banks of RAM that weren't ours. When our demos booted, they booted fast and looked great! But they were more fragile and a pain to maintain. We were optimizing for the end-user experience... fastest boot possible from a 3.5" floppy. Even I eventually moved away from that style of development during my last year of active IIgs usage (around 1992-93) when I finally got a 44MB SyQuest Hard Drive, and started writing ProDOS 8 programs and using some of the Memory Manager and sound tools.

Anyway, you should probably write everything for GSOS now. Yes, a bootable GSOS disk means you have very little space for a single disk release, but most people will probably experience a new release in emulation first... with a hard drive. And so many of us enthusiast have a mass storage device of some sort now, SCSI/CFFA/etc.

Also, don't be a dummy and go overboard with the "No Tools" philosophy like I did as a teenager. Just use the minimum amount needed to "play nice" with the OS... so you don't crash it and can exit your game/demo safely. Once you have allocated memory, and loaded your files, you can write all of the fun stuff directly to the hardware and have little interaction with the OS or tools. Basically exactly what Olivier said.

Excuse the rambling post, I'm just trying to get in this thread before I forget.

One last thing, soon in my video series, I'll cover the development environments needed to build these examples. In the meantime, if you already know how to use an assembler like Merlin, take a look at some examples I've given for building for a BLOAD-style binary, a ProDOS 8 program and a GSOS program that all just quit immediately. It's basically the exact same program written for 3 different environments.

https://github.com/digarok/gslib/blob/master/source/quit.s
https://github.com/digarok/gslib/blob/master/source/quit8.s

https://github.com/digarok/gslib/blob/master/source/quit16.s

So for example, the Quit16 program is basically the simplest "proper" GSOS app I think one can right. I'm not sure that last sentence is accurate though. For instance, you really would need to do some tool startup/shutdown routines in any real proper GSOS app, as shown in Olivier's original post. I'm just trying to show the simplest thing that will compile and run from GSOS. So don't flame me. I'm still just learning too! :lol:

```
         rel             ; compile as relocatable code
         dsk   Quit16.l    ; Save Name

         phk             ; Set Data Bank to Program Bank
         plb             ; Always do this first!

*** your main program here!!! ***

         jsl   $E100A8     ; Prodos 16 entry point
         da    $29         ; Quit code
         adrl  QuitParm    ; address of parameter table
         bcs   Error       ; never taken

Error    brk             ; should never get here

QuitParm adrl  $0000      ; pointer to pathname (not used here)
         da    $00         ; quit type (absolute quit)
```

Subject: Re: Tools or No Tools ?
Posted by jesseblue on Mon, 15 Jan 2018 19:34:21 GMT
View Forum Message <> Reply to Message

The answer is: No tools! 8)
