Introduction to GEOS Programming

ShadowM
ECCC 2015
Speaker Bio

- GEOS user since 1987
- several released applications including geoLink, geoSnap, ulecSwitch
- author of the infamous “Shadow Virus”
- GEOS pages at lyonlabs.org
Agenda

- familiarity with GEOS assumed
- knowledge of 6502 assembly assumed
- only "classic" geoProgrammer; no 128
- geoProgrammer build process
- using the main GEOS APIs
- demo program used for illustration
Learning Resources

You can also learn a lot from Maciej Witkowiak's disassembly of the GEOS kernel.

Make sure to check out my PRG errata sheet.

The Hitchhiker's Guide to GEOS

A Potpourri of Technical Programming Notes
(provided "as is" without support)

April 1988
geoProgrammer

- make sure to use 1.1
- geoAssembler (macro assembler)
- geoLinker (linkage editor)
- geoDebugger (symbolic debugger)
- 400-page manual
geoAssembler

- pseudo-ops, conditional assembly, macros
- local labels for branches
- pseudo-registers for zero-page locations
- source files are geoWrite documents
- bitmaps can be pasted into source
- errors written to a geoWrite file
- does not produce traditional listings
geoLinker

directives in geoWrite file
specifies whether SEQUENTIAL or VLIR
produces executable and symbol file(s)

```
.output          geosDemo
.header          geosDemoH.rel
.seq             geosDemoS.rel
geosDataS.rel
```
geoDebugger

» you really want to use an REU
» RESTORE hotkeys into debugger
» F7 to display hi-res screen and back again
» enter addresses as symbols in commands
» has its own macro language

.macro sc ;"show coordinates"
print"top/bottom: ",@r2L:,@r2H:.[cr]
print"left/right: ",@r3:,@r4:.[cr]
.endm
Files Needed to Create an Executable

- assembly source files
- include files (e.g. geosSym, geosMac)
- GEOS header file
- linkage directives file

Output files:
- executable program file
- debugger symbol table
- geoWrite symbol table (if requested)
Typical Program Initialization

- clear screen
- initialize menus, icons
- draw initial screen
- rts (to MainLoop), wait for events
Interrupt vs. MainLoop (simplified)

during interrupt handler:
- update mouse position and click status
- scan keyboard and populate queue
- update process/sleep timers

during MainLoop:
- service clicks: menus, icons, “otherPress”
- service keyboard events (e.g. GetString)
- service process/sleep timeouts
Planning Your GUI

Sorry folks, no layout manager!
Clearing the Screen

- SetPattern
- Rectangle vs. FrameRectangle
- background screen / RecoverRectangle

```assembly
lda  #2 ;50% stipple
jsr  SetPattern
LoadB r2L,0
LoadB r2H,199
LoadW r3,0
LoadW r4,319
jsr  Rectangle ;clear screen
```
Setting up Menus

sizing: trial & error, geoPaint, ruler DA

LoadW r0, mainMenu
lda #0
jsr DoMenu

mainMenu: .byte 0,14
          .word 0,61
          .byte HORIZONTAL | 2
          ;
          .word geosText
          .byte SUB_MENU
          .word geosMenu
          ;
          .word demoText
          .byte SUB_MENU
          .word demoMenu
          ;
geosText: .byte "geos",0
demoText: .byte "demo",0

[submenu definitions...]
Setting up Icons

- create your own in geoPaint
- system icons use University 12pt bold
- always call DoIcons, even if no icons!

LoadW r0,frzIcons
jsr DoIcons

frzIcons: .byte 1 ;number of icons
       .word 296 ;X-pos. to leave cursor
       .byte 7  ;Y-pos. to leave cursor

frzBmp: .word frzIcon ;address of bitmap
       .byte 34 ;X-position in bytes (left)
       .byte 0  ;Y-position in pixels (top)
       .byte 6,15 ;size (X: cards, Y: pixels)
frzPtr: .word svcRtn ;address of service routine
Dialog Boxes

- more than modal: state saved & restored
- DBTXTSTR, DBICON, DBGETFILES...

```assembly
LoadW r0, badPntDB
jsr DoDlgBox

badPntDB: .byte DEF_DB_POS | 1
        ;
        .byte DBTXTSTR
        .byte TXT_LN_X ; 16 pixels
        .byte TXT_LN_2_Y ; 32 pixels
        .word badPtMsg
        ;
        .byte OK
        .byte DBI_X_2 ; 17 cards
        .byte DBI_Y_2 ; 72 pixels
        .byte 0
        ;
badPtMsg: .byte "Invalid point size.", 0
```
String Handling

- PutString, GetString (baseline vs. top)
- string escapes for styles, location...

loading a font:

```
LoadW    r11,#XPOS
LoadB    r1H,#YPOS
LoadW    r0,string
jsr      PutString
```

```
LoadW    r0,fontName
jsr      OpenRecordFile
lda      #12
jsr      PointRecord
LoadW    r2,$6000-fontLoad
LoadW    r7,fontLoad
jsr      ReadRecord
LoadW    r0,fontLoad
jsr      LoadCharSet
```
### Bitmap Display

- Can be decompacted from memory or disk
- Static bitmaps can be pasted into source
- Assembler assigns values of `picW`, `picH`

```assembly
LoadW r0, bitmap1
LoadB r1L, #XPOS ; in cards
LoadB r1H, #YPOS ; in pixels
LoadB r2L, bitmap1W ; in cards
LoadB r2H, bitmap1H ; in pixels
jsr BitmapUp
```

```plaintext
bitmap1:
```

```
bitmap1W == picW
bitmap1H == picH
```

Processes

- "cooperative multi-threading"
- process handlers called during MainLoop
- call EnableProcess to start a process
- processes can be frozen or blocked

```
LoadW r0, procTbl
lda #NUM_PROC
jsr InitProcesses

procTbl: .word showRct
         .word 15
         .word showGfx
         .word 30
         .word showStr
         .word 45
```
DEMO PROGRAM
Resources

my GEOS page:
http://www.lyonlabs.org/commodore/onrequest/geos.html

- operating system, apps, programming tools
- reference manuals, programming “tips ‘n’ tricks” page
- reverse-engineered GEOS source (Maciej Witkowiak)
- source code for some apps I wrote

“Commodore GEOS” Google group

## geos IRC channel on Freenode