KAOS Toolkit Prototype

The KAOS Toolkit is a set of tools for use in development of applications for the Color Computer 3. The main focus of the prototype tools is to explore the details of converting and compiling of various multimedia assets into a format usable in retro computer game development. The Toolkit release currently includes three of the primary command line based tools; a palette converter, a general texture converter, and a texture compiler.

KAOS Palette Processor (KAOSPP)

The Palette Processor loads GIMP format palette files, checks the color values, and converts them to xxRGBRGB format, and generates an assembly language file containing a series of FCB statements that can be sent to the TC1014 (GIME) palette registers. The Palette Processor tool accepts the following arguments

--palette	Specifies the GIME palette file to convert. This option is required.
--format	Selects the target color format. The only value accepted by this argument is *tc1014.* If this option is not supplied the default format is *tc1014*.
--output-file	Specifies the filename to generate containing the converted palette. This option is required.

Example usage:

```
kaospp.exe --palette=palette.gpl --format=tc1014 –output-file=palette.asm
```

This will convert the file `palette.gpl` into TC1014 format and outputs the following assembly language code in `example.asm`.

```
FCB   $00,$07,$38,$0E,$0A,$1D,$39,$02,$10,$20,$24,$22,$34,$28,$2D,$3F
```

The palette processor does not provide any compensation for loss of precision during the conversion process and requires the R, G, and B color levels be 0, 85, 170, or 255. If the KAOS Palette Processor encounters an invalid channel value it will abort the conversion and display an error message.

KAOS Texture Processor (KAOSTP)

The Texture Processor provides basic features for converting a wide range of modern and Color Computer 3 specific image formats to a raw uncompressed image file. The texture processor can load and convert BMP, GIF, JPEG, PBM, PNG, TIFF, TGA, WEBP, MGE, VEF, CM3, and RAT image file formats. Regardless of the image format converted, all images must be 16 colors and should not contain any alpha. Regardless of what value the alpha component is set to the red, green, and blue color components are still used as-is.

The Palette Processor tool accepts the following arguments

--texture	Specifies the image file to convert. This option is required.
--palette	Selects the palette file used to convert RGB colors to 4 bit pixels matching the palette index of the color. This option is required.
--output-file	Specifies the filename to save the converted image to. This option is required.

Example usage:

```
kaostp.exe --palette=palette.gpl --texture=texture.png –output-file=texture.raw
```

This will convert the file `texture.png` into an uncompressed 4 bit per pixel image in packed pixel format. The palette is not saved with the converted image.

KAOS Texture Compiler (KAOSTC)

The Texture Compiler, commonly referring to as a sprite compiler, loads and converts image files in much the same way the Texture Processor does except that it generates directly executable Motorola MC6809 assembly language to draw the texture. The Texture compiler supports all image formats allowed by the Texture Processor plus the Piskel multi-frame sprite format. The Texture Compiler supports the following options.

--texture	The filename of the texture to compile. This option is required.
--palette	The palette to map pixel colors to. This option is required.
--label-prefix	The label prefix to use when generating assembly code. This option is required.
--pitch	The pitch or number of bytes per row. This option is required.
--stream-window-length	The number of bytes in a texture row processed by the compiler in a single optimization pass. The default value is 13.
--cursor-register	The main register to use as the cursor register containing the destination address to render to. Allowed values are X, Y, and U. This option is required.
--output-file	The name of the file generated containing the executable texture code. This option is required.
--variations	Variation type `Even` for even pixels alignment. `Odd` for odd pixel alignment. `Both` for even and odd pixel alignment. The default setting is `even`
--generate-exports	Generates label export statements in assembler output.
--restore-cursor	Generates a PULS instruction instead of a simple RTS to restore the cursor register and return when drawing is complete.
--bpp	The number of bits per pixel (1, 2, 4) on the target platform. This option is required.
--spritesheet	Specifies the input texture is a sprite sheet
--sheet-cell-width	The width of each cell in the sprite sheet
--sheet-cell-height	The height of each cell in the sprite sheet
--sheet-columns	Specifies the number of columns in a sprite sheet.
--sheet-rows	The number of rows in the sprite sheet to process
-sheet-margin	The outer margin of the sprite sheet

Example usage:

```
KAOSTC.exe   --palette=palette.gpl --label-prefix=MySprite_ --pitch=128 -bpp=4      \
             --cursor-register=Y --restore-cursor -texture=sprite1.png              \
             --output-file=sprite1.asm
```

This will compile the file `sprite1.png` as a 4 bits per pixel image mapping the colors to `palette.gpl` and generate MC6809 code to draw the image to a 128 byte wide 16 color frame buffer (256 pixel width) and use the Y register as the location to draw. See the sample output at the end of this section.

Unlike the Texture Processor tool the Texture Compiler processes the alpha channel. If the alpha channel is not set to the maximum value (255) that pixel is considered to be fully transparent and will not be drawn by the compiled sprite code.

Cursor Register

The Texture Compiler allows you to select which register is used to point to the frame buffer for drawing. Since how and where a sprite is drawn is controlled by other code which calculates where in the frame buffer to draw the Texture Compiler allows selecting any of the 3 available indexable registers on the MC6809 CPU – X, Y, and U. It is recommended that all efforts are made to use the Y register as the cursor since all load and store instructions for that register take an additional CPU cycle and the cursor register is only ever modified through a LEA instruction which uses the same number of cycles for all indexable registers.

Additionally the Texture Compiler provides two methods of returning after drawing is complete. The first is a simple RTS instruction and the second is through the use of a PULS instruction. The use of PULS was introduced to allow optimizations to the KAOS rendering pipeline as the cursor register it uses is also the primary index register for data access throughout a KAOS application.

Variations

The Texture compiler supports compiling the texture as-is as well as automatically generating an odd-pixel variation of the sprite. This shifts the sprite to the right by 1 pixel and generates an additional chunk of code allowing for pixel level positioning of a sprite. Odd-pixel variations generally require more CPU time to draw due to additional masking that may be introduced by the 1 pixel shift.

Stream Window Length

During compilation of larger sprites each row may be split up into multiple sections. This is required in order to reduce compile time due to the analysis of multiple paths of code generation. Without this window present some sprites could not be built in a reasonable amount of time. It is recommended that the default value of 13 is used for all sprites.

Sprite Sheets

The KAOS Texture Compiler also supports sprite sheets – images with multiple sprites arranged in a grid pattern.

The sprite sheet related options allow you to specify the size of the sprites, any margins around the edge of the entire image, along with the number of rows and columns to process. The example sheet shown above has 6 sprites each at 8x8 pixels, no margins on the sheet, 1 row, and 6 columns. This feature is relatively new to the prototype and has not gone through extensive testing and there are some features currently not provided such as specifying spacing between each sprite. This feature is intended to help migrate existing sprite sheets may have been created with Color Computer 3 programs such as ColorMax Deluxe and sheets exported from modern applications like Piskel.

Code Generation and Optimization

This version of the Compiler only supports left to right and top to down processing of the image and does not rely on any form of stack blasting. During the prototyping stage it was decided to take only the left-right/top-down approach to provide the most optimal code generation for small to medium sprites of all kinds and medium to large sprites with larger portions of transparent area and implement stack blasting in the final implementation. It accomplishes maximum speed by analyzing every possible code path utilizing all registers of the MC6809 processor. This means that the larger the image the longer the compiling process will take. Due to some limitations, images with a width greater than 26 pixels (13 bytes) may not receive full optimizations due to the windowing method used for code-path generation.

Sample output from sprite1.png

The following is the code generated from the example command line usage.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Texture filename: sprite1.png
; Palette filename: palette.gpl
; Pitch: 128 bytes
; Total cycles: 198
; Total bytes: 105
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MySprite_               ; 0  A=??/B=??   D=????    X=????    U=????    Y=0000
; NEW LINE
        lda     ,y      ; 4  A=??/B=??   D=????    X=????    U=????   [Y=0000]
        anda    #$0F    ; 2  A=??/B=??   D=????    X=????    U=????   [Y=0000]
        ldb     #$0C    ; 2  A=??/B=0C   D=????    X=????    U=????   [Y=0000]
        std     ,y      ; 5  A=??/B=0C   D=????    X=????    U=????   [Y=0000]
; NEW LINE
        leay    128,y   ; 8  A=??/B=0C   D=????    X=????    U=????   [Y=0080]
        ldd     #$10BB  ; 3  A=10/B=BB   D=10BB    X=????    U=????   [Y=0080]
        std     ,y      ; 5  A=10/B=BB   D=10BB    X=????    U=????   [Y=0080]
        lda     #$1B    ; 2  A=1B/B=BB   D=1BBB    X=????    U=????   [Y=0080]
        ldb     3,y     ; 5  A=1B/B=??   D=????    X=????    U=????   [Y=0080]
        andb    #$0F    ; 2  A=1B/B=??   D=????    X=????    U=????   [Y=0080]
        orb     #$10    ; 2  A=1B/B=??   D=????    X=????    U=????   [Y=0080]
        std     2,y     ; 6  A=1B/B=??   D=????    X=????    U=????   [Y=0080]
; NEW LINE
        leay    128,y   ; 8  A=1B/B=??   D=????    X=????    U=????   [Y=0100]
        ldd     #$B1B0  ; 3  A=B1/B=B0   D=B1B0    X=????    U=????   [Y=0100]
        std     ,y      ; 5  A=B1/B=B0   D=B1B0    X=????    U=????   [Y=0100]
        lda     #$BC    ; 2  A=BC/B=B0   D=BCB0    X=????    U=????   [Y=0100]
        ldb     3,y     ; 5  A=BC/B=??   D=????    X=????    U=????   [Y=0100]
        andb    #$0F    ; 2  A=BC/B=??   D=????    X=????    U=????   [Y=0100]
        std     2,y     ; 6  A=BC/B=??   D=????    X=????    U=????   [Y=0100]
; NEW LINE
        ldd     #$1B1B  ; 3  A=1B/B=1B   D=1B1B    X=????    U=????   [Y=0100]
        std     128,y   ; 9  A=1B/B=1B   D=1B1B    X=????    U=????   [Y=0100]
        ldb     #$10    ; 2  A=1B/B=10   D=1B10    X=????    U=????   [Y=0100]
        std     130,y   ; 9  A=1B/B=10   D=1B10    X=????    U=????   [Y=0100]
; NEW LINE
        leay    256,y   ; 8  A=1B/B=10   D=1B10    X=????    U=????   [Y=0200]
        ldd     #$0B01  ; 3  A=0B/B=01   D=0B01    X=????    U=????   [Y=0200]
        std     ,y      ; 5  A=0B/B=01   D=0B01    X=????    U=????   [Y=0200]
        clra            ; 2  A=00/B=01   D=0001    X=????    U=????   [Y=0200]
        ldb     3,y     ; 5  A=00/B=??   D=????    X=????    U=????   [Y=0200]
        andb    #$0F    ; 2  A=00/B=??   D=????    X=????    U=????   [Y=0200]
        orb     #$90    ; 2  A=00/B=??   D=????    X=????    U=????   [Y=0200]
        std     2,y     ; 6  A=00/B=??   D=????    X=????    U=????   [Y=0200]
; NEW LINE
        leay    128,y   ; 8  A=00/B=??   D=????    X=????    U=????   [Y=0280]
        ldd     #$1010  ; 3  A=10/B=10   D=1010    X=????    U=????   [Y=0280]
        std     ,y      ; 5  A=10/B=10   D=1010    X=????    U=????   [Y=0280]
        lda     #$BB    ; 2  A=BB/B=10   D=BB10    X=????    U=????   [Y=0280]
        ldb     3,y     ; 5  A=BB/B=??   D=????    X=????    U=????   [Y=0280]
        andb    #$0F    ; 2  A=BB/B=??   D=????    X=????    U=????   [Y=0280]
        orb     #$10    ; 2  A=BB/B=??   D=????    X=????    U=????   [Y=0280]
        std     2,y     ; 6  A=BB/B=??   D=????    X=????    U=????   [Y=0280]
; NEW LINE
        leay    129,y   ; 8  A=BB/B=??   D=????    X=????    U=????   [Y=0301]
        lda     ,y      ; 4  A=??/B=??   D=????    X=????    U=????   [Y=0301]
        anda    #$F0    ; 2  A=??/B=??   D=????    X=????    U=????   [Y=0301]
        ora     #$01    ; 2  A=??/B=??   D=????    X=????    U=????   [Y=0301]
        ldb     #$01    ; 2  A=??/B=01   D=????    X=????    U=????   [Y=0301]
        std     ,y      ; 5  A=??/B=01   D=????    X=????    U=????   [Y=0301]
; NEW LINE
; END
        puls    y,pc    ; 9  A=??/B=01   D=????    X=????    U=????   [Y=0301]
```