

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

Jakiś czas temu, w ramach potrzeby tworzenia filmików CDXL, napisałem skrypt ARexxa korzystający z dobrodziejstw kombajnu graficznego Adpro. Był to jednocześnie czas spędzony na nauce tego języka, ponieważ wcześniej go nie znałem. W związku z tym, chcę połączyć pożyteczne z nauką i dokładnie omówić ten skrypt. Korzysta on intensywnie z rozkazów Adpro, które są w pełni opisane w podręczniku do tego programu. Skrypt pod nazwą CreateXL wrzucamy do podkatalogu Commands2. Po uruchomieniu Adpro w okienku User Commands zobaczymy nazwę naszego skryptu. Po kliknięciu na przycisk skrypt uruchomi się. Rzecz jasna, wcześniej musi być uruchomiony RexxMast.

**** Create CDXL movie from cell pictures ** This script requires ADPro v2.5.0 (or higher).**

Każdy skrypt Arexxa musi zaczynać się komentarzem.

ADDRESS "ADPro";

Komenda ADDRESS ustawia aktualny port Arexxa, tutaj Adpro.

OPTIONS RESULTS

Każda funkcja Adpro zwraca wartość błędu do zmiennej nazwanej RC. Dzięki komendzie OPTIONS RESULTS, właściwa wartość funkcji przekazywana jest do zmiennej ADPRO_RESULT. Dlatego też, musi to być jedna z pierwszych komend użyta w skryptach Adpro. Druga rzecz, to wartość ADPRO_RESULT może być niezdefiniowana, jeśli wartość RC jest różna od zera (wystąpił błąd), dlatego najpierw sprawdzamy wartość RC.

NL = '0A'X SQ = '27'X /* ' */ DQ = '22'X /* " */ TRUE = 1 FALSE = 0

Tutaj definiujemy kilka zmiennych, niekoniecznie potrzebnych, ale często używanych w skryptach do Adpro. Liczba X oznacza liczbę w systemie hexadecymalnym.

ADPRO_TO_FRONT

Ta prosta komenda wyświetla ekran czy okno Adpro, jeśli było schowane.

TempDefaults ="T:TempADProDefaults"; SAVE_DEFAULTS TempDefaults

Nie chcemy zmieniać domyślnie zapisanych ustawień, więc je zapisujemy do tymczasowego pliku. W zasadzie powinniśmy sprawdzić jeszcze wartość RC, ale większość dołączonych skryptów do Adpro tego nie robi.

OBTAIN_ADPRO /* Exclusive "lock" */ IF (RC ~= 0) THEN DO ErrorText = 'ADPro in use'
CALL LabelErr END

Tutaj informujemy Adpro, że przejmujemy go do własnych celów ekskluzywnie, więc w tym czasie inne skrypty chcące używać Adpro nie będą wykonywane. Wiecie, multitasking. Tutaj już jest obsługa błędów. Komenda CALL jest w tym przypadku zamiennikiem basikowego Goto/Gosub. Należy też zauważyć, że można stosować zamiennie cudzysłowie pojedyncze i podwójne, ale bez mieszania.

CLOSE_RENDER_SCREEN CLEAR_RAW CLEAR_RENDERED

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

Te komendy wymagają omówienia sposobu pracy Adpro. Tenże program służący do obróbki obrazu pracuje wewnątrz w trybie 24-bitowym (lub 8-bitowym gray) nazwanym RAW. Każdy wgrany obrazek jest konwertowany do trybu raw, także obrazki indeksowane (2-256 kolorów, EHB, HAM). Tryb RENDERED jest konwersją z obrazu 24-bitowego do obrazu indeksowanego w tym HAM czy np. DCTV. Więc, gdy wgramy obrazek 24-bitowy to Adpro tworzy bufor typu raw. Gdy wgramy obrazek indeksowany to Adpro tworzy 2 bufor: raw i rendered. Możemy renderować obrazy 24-bitowe do pożądanego trybu ekranu wyświetlanego na Amidze. Po wyrenderowaniu Adpro otwiera pożądaną ekran w celu jego wyświetlenia. Jest on otwarty i schowany. Zastosowane komendy zamykają ten ekran i zwalniają bufor.

```
/* Use all available render modes (eg. AGA on OCS) */ AVAIL_MODES_ONLY_OFF
```

Ta komenda pozwala na renderowanie obrazów w trybach niedostępnych dla używanego chipsetu (e domyśle tryby AGA na chipsecie ECS). Oczywiście, chodzi o rendering, nie wyświetlanie.

```
/* Palette unlocked & aga */ PSTATUS UNLOCKED PWIDTH ENHANCED
```

System palety kolorów jest w Adpro całkiem rozbudowany. Paleta może być zablokowana, niestety tylko globalnie. Naturalnie odblokowujemy. Następnie mamy 2 tryby rozdzielczości palety: normalny i rozszerzony. Ten drugi generuje kolory dokładniej. Naturalny domysł sugeruje, że paleta enhanced jest 24-bitowa dla układów AGA, zaś normal to paleta 12-bitowa dla ECS. Niestety nie i to generuje szereg problemów z CDXL, ale o tym później. Różnica jest, ale raczej nie tak duża, pewnie chodzi o dokładność obliczeń. Generalnie nie można mieć pełnej kontroli nad obrazem wyrenderowanym dla chipsetu OCS/ECS. Ujmę to przykładem: generujemy obrazek w 32 kolorach, ale w odcieniach szarości. Bez względu na rozdzielczość palety (oraz inne przełączniki), obrazek zostanie wygenerowany w 32 odcieniach szarości na układach ECS. Po jego wyświetleniu rzeczywiście jest tych odcieni 16, ale edytować palety w 12-u bitach nie można. Dlatego też dajemy enhanced bez względu na chipset.

```
RENDER_TYPE 256 /* Remove Custom Palette */
```

Ta komenda ustawia rendering na 256-kolorów. Po co, skoro nie wybraliśmy jeszcze trybu? Ano dlatego, bo istnieje opcja Custom Palette, która może zaburzyć działanie skryptu (może być włączona przed uruchomieniem skryptu). Włączenie trybu renderingu do pełnej liczby kolorów wyłącza jednocześnie tryb CUST. Sam tryb Custom Palette służy do renderowania wybranej grupy kolorów np. tworzymy obrazek 5-kolorowy na ekranie 8-kolorowym zaczynając przykładowo od drugiego indeksu. Jest to przydatne w grach. Sam tryb CUST nie mówi jaki mamy tryb, więc trzeba ustawić jakiś, potem dopiero CUST. Na teraz ustawiamy jakiś.

```
/* default strings: error, interleaved, render */ ErrorText = 'Finished' ivd = " it = 'IMAGE'
```

Teraz ustawimy kilka domyślnych zmiennych. Będą użyte później. ivd na razie jest pustym ciągiem, będzie użyty jako parametr dla innej funkcji. Podobnie it, ma specyficzną wartość, też jest parametrem.

```
/* Get long string with sum of filenames, beware if too long */ GETFILES '"Select A Cell Frames"' IF (RC ~= 0) THEN DO ErrorText = 'Cancelled' CALL LabelErr END TheFiles = Trim(ADPRO_RESULT)
```

Adpro udostępnia kilka funkcji do obsługi wyboru plików na bazie biblioteki ASL. Jako, że używam patchy do Reqtools, zastosowałem GETFILES, które niejako wymusza zaznaczanie przyciskiem ALL. W przypadku ASL nie ma przycisku ALL, więc może być trudno, bo nie wiem czy jest jakiś klawiaturowy skrót. Dla ASL należałoby inaczej skonstruować wybieranie plików np. na zasadzie katalogu lub nazwy bazowej. W funkcji GETFILES istnieje poważny problem.

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

ADPRO_RESULT przyjmie wartość ciągu tekstowego: "filename1" "filename2" "filename3" itd. Ciąg ma jednak ograniczoną długość, więc przy dużej ilości plików skrypt się powiesi. Przy około 2000 plików jest dobrze, ale trzeba o tym ograniczeniu pamiętać. Funkcja Trim usuwa spację z końca ciągu, gdyż ciąg jest konstruowany na zasadzie nazw w cudzysłowach z dodaną spacją na końcu, a ostatnia spacja nie jest nam potrzebna. W zasadzie funkcja Trim nie jest niezbędna.

```
NrOfFiles = 0 DO WHILE TheFiles ~= "          /* create substring with one filename */          PARSE VAR  
TheFiles Filename '&quot; &quot;' TheFiles          /* move substring to array (with removing blanks & &quot; */  
          File.NrOfFiles= STRIP(Filename,B,'&quot;')          /* check to exists file */          IF EXISTS(  
File.NrOfFiles ) = TRUE THEN NrOfFiles = NrOfFiles + 1 END IF (NrOfFiles = 0) THEN DO          ErrorText= 'No  
Source Files'          CALL LabelErr END
```

Ten kawałek kodu jest bardziej skomplikowany. Generalnie tworzy tablicę z nazwami plików, zlicza ich liczbę oraz sprawdza czy pliki rzeczywiście istnieją. Indeks tablicy nadawany jest po kropce, inaczej niż w większości języków. Użyto tutaj dość trudnej w zrozumieniu funkcji PARSE VAR. Działa to zasadzie wycinania subciągu (tutaj Filename) z ciągu (tutaj pierwsze TheFiles) i przekazanie reszty do kolejnej zmiennej (tutaj ta sama zmienna TheFiles). Skąd wiadomo ile wyciąć? Służy ku temu parametr z podanymi ogranicznikami (tutaj " "). Działanie pętli powoduje, że tablica wypełnia się nazwami pojedynczych plików, zaś ciąg TheFiles zmniejsza się aż do pustego ciągu. Problemem jest nazwa pierwszego pliku i ostatniego. Obie zawierają dodatkowy cudzysłów. Usuujemy go funkcją STRIP, która usuwa także spacje z obu końców. Spacje w środku są dozwolone. Przykładowo ciąg TheFiles: >>"file1" "file2" "file3"<< po pierwszej iteracji zredukuje się do >>file2" "file3"<< zaś Filename będzie zawierać >>"file1<<. Po drugiej iteracji TheFiles = >>file3"<<, zaś Filename >>file2<<. Po kolejnej, TheFiles będzie pusty, zaś Filename >>file3"<<. Chyba wszystko jasne.

```
GETFILE '&quot;Select A CDXL File&quot;' IF (RC ~= 0) THEN DO          ErrorText= 'Cancelled'          CALL  
LabelErr END XLFile = ADPRO_RESULT
```

Tutaj już prosto, wybieramy plik docelowy. GETFILE pozwala wybrać tylko jeden plik w odróżnieniu od GETFILES.

```
/* check for input & output filenames */ FileNr = 0 DO WHILE (FileNr &lt; NrOfFiles)          IF(File.FileNr =  
XLFile) THEN DO          ErrorText = 'Error, the same input & output'          CALL LabelErr  
          END          FileNr= FileNr + 1 END
```

Teraz sprawdzamy czy plik docelowy nie jest jednym ze źródłowych.

```
/* always create a new cdxl file, no append */ IF EXISTS( XLFile ) = TRUE THEN DO          OKAY2 'File exists.  
Delete?'          IF RC = 1 THEN ADDRESS COMMAND '&quot;Delete &gt;NIL:&quot; DQ || XLFile || DQ          IF  
EXISTS( XLFile ) = TRUE THEN DO          ErrorText = 'Delete Error'          CALL LabelErr  
          END END
```

Saver CDXL potrafi dogrywać nowe klatki do utworzonego już pliku, ale muszą zgadzać się wszystkie parametry obrazu. Zablokowałem tę możliwość z pewnych względów i dlatego skrypt sprawdza czy już taki plik istnieje. Jeśli tak to pojawia się requester z pytaniem o jego usunięcie. Służy do tego komenda OKAY2, która zwraca numer przycisku OK lub Cancel do zmiennej RC. ADDRESS COMMAND używa programu z CLI do usunięcia pliku. Wtedy mamy pewność, że tworzymy całkiem nowy plik CDXL.

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

OKAYN '"CreateXL","Choose type of operation",' '"Scale & Render|Render|Join Only",' xl = RC

OKAYN wyświetla requester z kilkoma przyciskami do wyboru. Zmienna xl zawiera wartość zależną od wyboru. Scale & Render powoduje, że nasze obrazki będą skalowane i renderowane, przy samym Render nie będzie skalowania. Join Only, tylko połączy pliki. Jako, że tutaj rendering nie będzie używany, to obrazki muszą być przygotowane w jednakowym trybie i rozmiarze. Ta ostatnia opcja jest dla tych co wolą używać innych programów do renderowania.

OKAY2 '"NonInterleaved?",' IF RC = 0 THEN ivd = 'INTERLEAVED'

Tutaj pytamy się czy film CDXL ma być w formacie interleaved. Domyślnie jest, że nie, dlatego pytanie jest odwrotne. Po prostu cdsjsxl nie suportuje, zaś Xlplay jak najbardziej. Np. na CD-ROM z dinozaurami jest sporo takich filmików.

```
/* if render */ IF xl ~= 0 THEN DO          RenderList = '&quot;HAM&quot; &quot;2&quot; &quot;4&quot;
&quot;8&quot; &quot;16&quot; &quot;32&quot; &quot;64&quot; &quot;128&quot; &quot;256&quot;
&quot;EHB&quot; &quot;HAM&quot; &quot;HAM8&quot; &quot;GRAY&quot; &quot;24-bit&quot;'          LISTVIEW
'&quot;Render Depth Available:&quot;' 15 ITEMS RenderList          IF (RC = 0) | (RC = 1) THEN PARSE VAR
ADPRO_RESULT '&quot;'RenderStr&quot;,&quot;scratch          ELSE DO          ErrorText = 'Cancelled'
          CALL LabelErr          END          /* for 24-bit rendering not needed */          IF ( RenderStr =
'24-bit' ) | ( RenderStr = 'GRAY') THEN it = 'RAW'          ELSE RENDER_TYPE RenderStr          /* if rendering
*/          IF it ~= 'RAW' THEN DO          DitherList = '&quot;Floyd&quot; &quot;None&quot;
&quot;Floyd&quot; &quot;Burkes&quot; &quot;Sierra&quot; &quot;Jarvis&quot; &quot;Stucki&quot;
&quot;Random&quot; &quot;Lg Ord&quot; &quot;Sm Ord&quot;'          LISTVIEW '&quot;Dithers
Available:&quot;' 10 ITEMS DitherList          IF (RC = 0) | (RC = 1) THEN PARSE VAR ADPRO_RESULT
'&quot;'DitherStr&quot;,&quot;scratch          ELSE DO          ErrorText = 'Cancelled'
          CALL LabelErr          END          DitherMode = 0 /* default */          IF
(DitherStr = '&quot;Floyd&quot;') THEN DitherMode = 1          ELSE IF (DitherStr = '&quot;Burkes&quot;')
THEN DitherMode = 2          ELSE IF (DitherStr = '&quot;Sierra&quot;') THEN DitherMode = 3
          ELSE IF (DitherStr = '&quot;Jarvis&quot;') THEN DitherMode = 4          ELSE IF (DitherStr =
&quot;Stucki&quot;') THEN DitherMode = 5          ELSE IF (DitherStr = '&quot;Random&quot;') THEN
DitherMode = 6          ELSE IF (DitherStr = '&quot;Lg Ord&quot;') THEN DitherMode = 7
          ELSE IF (DitherStr = '&quot;Sm Ord&quot;') THEN DitherMode = 8          DITHER DitherMode
          IF (DitherMode = 6) | (DitherMode = 7) | (DitherMode = 8) THEN DO          DitherAmt = 16 /*
default */          GETNUMBER '&quot;Enter Dither Amount&quot;,' 16 1 256          IF
(RC = 0) THEN DitherAmt = ADPRO_RESULT          DITHER_AMOUNT DitherAmt
          END          /* get nr of colors for render type */          PTOTAL          palette
= ADPRO_RESULT /* 2..256, EHB, HAM */          cols = palette          IF palette = 'EHB' THEN
cols = 32          IF palette = 'HAM' THEN cols = 16          IF palette = 'HAM8' THEN cols = 64
          IF cols > 2 THEN DO          OKAYN '&quot;CreateXL&quot;,&quot;Use zero
color?&quot;,' '&quot;Yes|Not (for genlock)&quot;
          PPOKE 0 0 0 0 /* set color zero to black */          SETPALINFO palette cols-pal pal /*
set/unset color zero &quot;lock&quot; */          END          OKAYN '&quot;CreateXL&quot;,'
'&quot;Choose type of palette&quot;,' '&quot;One per frame|One per movie&quot;,'          /* one palette per
movie must be loaded from disk */          IF RC = 0 THEN DO          PLOAD /* for genlock
load truncated palette */          IF (RC ~= 0) THEN DO          ErrorText =
'Load Error'          CALL LabelErr          END          /* lock
palette for all frames */          PSTATUS LOCKED          END          END END
```

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

Ten fragment programu jest wykonywany tylko w przypadku wyboru renderowania. Wywołuje szereg zapytań i ustawia odpowiednie parametry. Powyższy fragment będę omawiał kawałkami dla lepszego zrozumienia.

```
RenderList = '&quot;HAM&quot; &quot;2&quot; &quot;4&quot; &quot;8&quot; &quot;16&quot;&quot;32&quot;
&quot;64&quot; &quot;128&quot; &quot;256&quot; &quot;EHB&quot;&quot;HAM&quot; &quot;HAM8&quot;
&quot;GRAY&quot; &quot;24-bit&quot;,' LISTVIEW '&quot;Render Depth Available:&quot;,' 15 ITEMS RenderList IF (RC
= 0) | (RC = 1) THEN PARSE VAR ADPRO_RESULT '&quot;RenderStr&quot;,'scratch ELSE DO          ErrorText =
'Cancelled'          CALL LabelErr END
```

Komenda Adpro LISTVIEW wyświetla listę z wyborem. Używamy komendy PARSE VAR do wyodrębnienia ciągu z nazwą trybu renderowania, który przekazany zostaje do zmiennej RenderStr. Nie używam tutaj wyboru trybu ekranu z tego względu, że filmy CDXL nie zawierają w sobie takiej informacji. Jest tylko flaga HAM oraz flaga Advanced Video Mode (równoznaczna z Hires), ale tej ostatniej Saver CDXL nie suportuje. Więc faktycznie wszystkie filmy będą w Lowres. Tryb EHB jest zapisywany z pełną paletą 64 kolorów, więc rozpoznanie takiego pliku jest nieco (ale tylko nieco) problematyczne. Dwa tryby: GRAY i 24-bit. Są to tryby chunky. Format CDXL takowe przewiduje i Adpro takowe tworzy i odczytuje. Przestrzegam jednak przed tworzeniem takich plików, nie ma do nich odtwarzacza. Z drugiej strony dodanie ich komplikuje dość mocno ten skrypt.

```
/* for 24-bit rendering not needed */ IF (RenderStr = '24-bit') | ( RenderStr = 'GRAY') THEN it = 'RAW' ELSE
RENDER_TYPE RenderStr
```

Owe tryby chunky są faktycznie obrazami Adpro w formacie raw, czyli są tworzone automatycznie. Koniec końców nie jest to rendering. Z tego względu flaga it przyjmuje wartość RAW (wcześniej IMAGE). W pozostałych przypadkach ustawiamy wybrany RENDER_TYPE. Poniższa część dotyczy już rzeczywistego renderingu.

```
DitherList = '&quot;Floyd&quot; &quot;None&quot; &quot;Floyd&quot; &quot;Burkes&quot;&quot;Sierra&quot;
&quot;Jarvis&quot; &quot;Stucki&quot; &quot;Random&quot;&quot;Lg Ord&quot; &quot;Sm Ord&quot;,' LISTVIEW
'&quot;Dithers Available:&quot;,' 10 ITEMS DitherList IF (RC = 0) | (RC = 1) THEN PARSE VAR ADPRO_RESULT
'&quot;DitherStr&quot;,' scratch ELSE DO          ErrorText = 'Cancelled'          CALL LabelErr END
DitherMode = 0 /* default */ IF (DitherStr = &quot;Floyd&quot;) THEN DitherMode = 1 ELSE IF (DitherStr =
&quot;Burkes&quot;) THEN DitherMode = 2 ELSE IF (DitherStr = &quot;Sierra&quot;) THEN DitherMode = 3 ELSE IF
(DitherStr = &quot;Jarvis&quot;) THEN DitherMode = 4 ELSE IF (DitherStr = &quot;Stucki&quot;) THEN DitherMode = 5
ELSE IF (DitherStr = &quot;Random&quot;) THEN DitherMode = 6 ELSE IF (DitherStr = &quot;Lg Ord&quot;) THEN
DitherMode = 7 ELSE IF (DitherStr = &quot;Sm Ord&quot;) THEN DitherMode = 8 DITHER DitherMode IF (DitherMode
= 6) | (DitherMode = 7) | (DitherMode = 8) THEN DO          DitherAmt = 16 /* default */          GETNUMBER
'&quot;Enter Dither Amount&quot;,' 16 1 256          IF (RC = 0) THEN DitherAmt = ADPRO_RESULT
DITHER_AMOUNT DitherAmt END
```

Tutaj wybieramy dithering. Jako, że komenda Adpro DITHER wymaga argumentu liczbowego, przypisujemy DitherMode odpowiednio do DitherStr. Niektóre typy ditheringu wymagają podania wielkości tablicy. Służ do tego komenda GETNUMBER.

```
/* get nr of colors for render type */ PTOTAL palette = ADPRO_RESULT /* 2..256, EHB, HAM */ cols = palette IF
palette = 'EHB' THEN cols = 32 IF palette = 'HAM' THEN cols = 16 IF palette = 'HAM8' THEN cols = 64
```

Funkcja PTOTAL zwraca wartość równa maksymalnej ilości kolorów dla danego trybu renderowania. Problemem jest,

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

że w specjalnych przypadkach (EHB, HAM) zwraca nazwę trybu, a nie ilość kolorów. Odpowiednio reagujemy.

```
IF cols > 2 THEN DO          OKAYN '"CreateXL"' '"Use zero color?"' '"Yes|Not (for
genlock)"'          pal = 1 - RC /* if use pal=0 */          PPOKE 0 0 0 0 /* set color zero to black */
SETPALINFO palette cols-pal pal /* set/unset color zero "lock" */ END
```

Tu pytamy czy przy renderowaniu będzie używany kolor zerowy. Po co? Jeśli każda klatka ma inną paletę, to spowoduje miganie ramki (na OCS nie ma borderblank). Nie ma możliwości ustawiania i blokowania pojedynczych kolorów. Podane rozwiązanie jest połowiczne. Jeśli klikniemy nie, to kolor zerowy ustawiamy na czarno (komenda PPOKE). Komenda SETPALINFO ustawia zakres używanych kolorów: pełny lub niepełny bez koloru zero (co jednocześnie jest powodem do zmiany trybu na CUST, później). Zmienna pal jest flagą. Oczywiście, powyższe zapytanie nie ma sensu dla trybu 2-kolorowego.

```
OKAYN '"CreateXL"' '"Choose type of palette"' '"One per frame|One per movie"' /*
one palette per movie must be loaded from disk */ IF RC = 0 THEN DO          PLOAD /* for genlock load truncated
palette */          IF (RC ~= 0) THEN DO          ErrorText = 'Load Error'          CALL LabelErr          END
/* lock palette for all frames */          PSTATUS LOCKED END
```

Ostatnie zapytanie dotyczy, czy paleta ma być różna co klatkę lub stała. Filmy CDXL zapisują paletę w każdej klatce zawsze, ale czasami może być przydatne użycie własnej. W zasadzie nie ma prostej możliwości samemu wygenerowania takiej palety w Adpro, więc opcja sprowadza się do wgrania już gotowej z dysku (można taką przygotować np. programem AnimLab), służy do tego komenda PLOAD (przesunięcie kolorów jest oczywiście uwzględniane). W tym przypadku paletę należy zablokować (PSTATUS).

```
/* main loop */ FileNr = 0 DO WHILE (FileNr < NrofFiles)          /* loader any format known by ADPro */
LOADER UNIVERSAL DQ || File.FileNr || DQ          IF (RC ~= 0) THEN DO          ErrorText = 'Load Error'
CALL LabelErr          END          FileNr = FileNr + 1          IF xl = 0 THEN DO /* join only */
IMAGE_TYPE          ct = ADPRO_RESULT          IF FileNr = 1 THEN DO          /* check first
frame type 24-bit or clut */          IF ct = 'COLOR' THEN it = 'RAW'          END          /* special
cases for image type gray */          IF it = 'RAW' THEN DO          IF WORD( ADPRO_RESULT, 1 ) =
'GRAY' THEN DO          /* must convert to 24-bit */          OPERATOR
GRAY_TO_COLOR          IF (RC ~= 0) THEN DO          ErrorText = 'Operation
Error'          CALL LabelErr          END          END          END
ELSE IF ct = 'GRAY' THEN DO          /* must render to 256 colors */          DITHER 0
/* no dithering */          RENDER_TYPE 256          EXECUTE          IF (RC ~= 0)
THEN DO          ErrorText = 'Render Error'          CALL LabelErr
END          END          END          ELSE DO          /* render */          IF xl = 1 THEN DO
/* scaling */          IF FileNr = 1 THEN DO          XSIZE          iw =
ADPRO_RESULT          YSIZE          ih = ADPRO_RESULT
GETNUMBER '"Enter Image Width"' iw 16 iw          IF (RC = 0) THEN iw =
ADPRO_RESULT          GETNUMBER '"Enter Image Height"' ih 16 ih
IF (RC = 0) THEN ih = ADPRO_RESULT          END          ABS_SCALE iw ih          END
/* GRAY is special image, rendering to HAM/EHB not possible */          ct = FALSE
IMAGE_TYPE          IF WORD( ADPRO_RESULT, 1 ) = 'GRAY' THEN DO          IF it = 'RAW' THEN
DO /* no need rendering */          IF RenderStr = '24-bit' THEN ct = TRUE          END
ELSE DO /* need rendering */          IF RenderStr = 'EHB' THEN ct = TRUE
IF RenderStr = 'HAM' THEN ct = TRUE          IF RenderStr = 'HAM8' THEN ct = TRUE
END          END          IF RenderStr = 'GRAY' THEN DO          IF WORD(
```

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

```
ADPRO_RESULT, 1 ) = 'GRAY' THEN DO                ct = FALSE                END
ELSE DO                OPERATOR COLOR_TO_GRAY                IF (RC ~= 0) THEN DO
    ErrorText = 'Operation Error'                CALL LabelErr                END
    END                END                /* if conversion needed */                IF ct = TRUE THEN DO
    OPERATOR GRAY_TO_COLOR                IF (RC ~= 0) THEN DO                ErrorText
= 'Operation Error'                CALL LabelErr                END                END                /*
rendering if not raw */                IF it = 'IMAGE' THEN DO                RENDER_TYPE RenderStr
    IF pal = 1 THEN RENDER_TYPE 'CUST'                EXECUTE                IF (RC ~= 0) THEN DO
    ErrorText = 'Render Error'                CALL LabelErr                END
END                END                IF FileNr = 1 THEN DO                pad = 0                GETNUMBER '&quot;Enter Pad
Space&quot;' 0 0 999                IF (RC = 0) THEN pad = ADPRO_RESULT                END                /* save cdxl frame,
noaudio, nopad */                SAVER CDXL DQ || XLFile || DQ it FRAMENUM FileNr PADSPACE pad SAVEPAD ivd
IF (RC ~= 0) THEN DO                ErrorText = 'Save Error'                CALL LabelErr                END END
```

Powyższa pętla jest "gwoździem" programu. Znowu rozbijemy ją sobie na czynniki proste.

```
/* loader any format known by ADPro */ LOADER UNIVERSAL DQ || File.FileNr || DQ IF (RC ~= 0) THEN DO
ErrorText = 'Load Error'                CALL LabelErr END
```

Komenda LOADER wgrywa obrazek do pamięci. Został wybrany UNIVERSAL, który zna większość formatów (nie ma PNG), ale można wpisać jakiś specyficzny. Forma DQ||File.FileNr||DQ dodaje cudzysłowy do nazwy pliku.

Poniżej omówimy części dotyczące "Join Only".

```
IMAGE_TYPE ct = ADPRO_RESULT IF FileNr = 1 THEN DO                /* check first frame type 24-bit or clut */                IF ct
= 'COLOR' THEN it = 'RAW' END
```

Funkcja IMAGE_TYPE zwraca rodzaj wgranego obrazka w postaci ciągu BITPLANE (obrazek indeksowany lub wyrenderowany), COLOR (obrazek RAW w kolorze), GRAY (obrazek RAW w odcieniach szarości), NONE (brak obrazu). Bardzo często ciąg może zawierać podwójne słowa: "COLOR BITPLANE" lub "GRAY BITPLANE", wynika to ze sposobu pracy Adpro o czym wspominałem na początku. Pierwszy plik definiuje rodzaj filmu CDXL jaki powstanie. Jeśli jest to obraz indeksowany to kolejne też muszą być i to identyczne. Jeśli jednak będzie to obrazek 24-bitowy to kolejne mogą być w dowolnej ilości kolorów, bo konwersja do 24-bit jest automatyczna. Dlatego w takim przypadku zmieniamy flagę it na RAW.

```
/* special cases for image type gray */ IF it = 'RAW' THEN DO                IF WORD( ADPRO_RESULT, 1 ) = 'GRAY' THEN
DO                /* must convert to 24-bit */                OPERATOR GRAY_TO_COLOR                IF (RC ~= 0) THEN
DO                ErrorText = 'Operation Error'                CALL LabelErr                END                END END
```

Tutaj sprawdzamy warunki wynikające z poprzedniego wątku. Pierwszy warunek dotyczy generowania filmów CDXL w 24-bitach. Może wtedy się zdarzyć, że któryś z kolejnych obrazków jest typem GRAY. Sprawdzamy to funkcją WORD, która wyszukuje te słowo w ciągu zwróconym przez IMAGE_TYPE. W takim przypadku musimy skonwertować obraz typu GRAY na obraz typu COLOR (oczywiście nie znaczy to, że obraz zostanie pokolorowany). Służy do tego komenda OPERATOR.

```
ELSE IF ct = 'GRAY' THEN DO                /* must render to 256 colors */                DITHER 0 /* no dithering */
RENDER_TYPE 256                EXECUTE                IF (RC ~= 0) THEN DO                ErrorText = 'Render Error'
```

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

```
CALL LabelErr      END END
```

W drugim warunku (wykluczającym się, bo tworzymy film indeksowany), może zdarzyć się, że któryś obrazek będzie typem "GRAY", a nie "COLOR BITPLANE" lub "GRAY BITPLANE". Dlaczego? Specyfika programu. Nas interesuje tylko typ BITPLANE, a może go brakować. Wtedy musimy go sobie wyrenderować. Wyłączamy dithering, ustawiamy typ renderingu. Komenda EXECUTE renderuje obrazek. Generalnie w przypadku Join Only nie jest możliwe tworzenie filmów 8-bit chunky gray.

W przypadku renderowania mamy do czynienia z opcjonalnym skalowaniem.

```
      IF xl = 1 THEN DO                      /* scaling */
XSIZE                                iw = ADPRO_RESULT      YSIZE      ih =
ADPRO_RESULT      GETNUMBER '&quot;Enter Image Width&quot;' iw 16 iw
IF (RC = 0) THEN iw = ADPRO_RESULT      GETNUMBER '&quot;Enter Image Height&quot;' ih 16
ih      IF (RC = 0) THEN ih = ADPRO_RESULT      END
ABS_SCALE iw ih      END
```

Dla pierwszego pliku wybieramy sobie wymiary docelowe filmu, przy czym dopuściłem tylko skalowanie w dół. Funkcje XSIZE i YSIZE zwracają rozmiar obrazu.

```
      /* GRAY is special image, rendering to HAM/EHB not possible */      ct = FALSE
IMAGE_TYPE      IF WORD( ADPRO_RESULT, 1 ) = 'GRAY' THEN DO      IF it = 'RAW' THEN
DO /* no need rendering */      IF RenderStr = '24-bit' THEN ct = TRUE      END
      ELSE DO /* need rendering */      IF RenderStr = 'EHB' THEN ct = TRUE
      IF RenderStr = 'HAM' THEN ct = TRUE      IF RenderStr = 'HAM8' THEN ct = TRUE
      END      END
```

Ten fragment sprawdza typ obrazu. Chodzi, o to, że Adpro nie renderuje do EHB/HAM obrazków typu GRAY. W takim przypadku trzeba zamienić obraz w typ COLOR, podobnie jak wcześniej. Podobną konwersję trzeba zrobić w przypadku pseudorenderingu do 24-bit. Flaga ct wskazuje czy konwersja jest potrzebna.

```
      IF RenderStr = 'GRAY' THEN DO      IF WORD( ADPRO_RESULT, 1 ) = 'GRAY' THEN DO
      ct = FALSE      END      ELSE DO
OPERATOR COLOR_TO_GRAY      IF (RC ~= 0) THEN DO      ErrorText =
'Operation Error'      CALL LabelErr      END      END
      END
```

Tutaj jest przypadek odwrotny. Gdy chcemy renderować do GRAY to używamy odpowiedniego operatora.

```
      /* if conversion needed */      IF ct = TRUE THEN DO      OPERATOR
GRAY_TO_COLOR      IF (RC ~= 0) THEN DO      ErrorText = 'Operation Error'
      CALL LabelErr      END      END
```

Tutaj konwertujemy do typu COLOR w zależności od potrzeby (flaga ct).

```
      /* rendering if not raw */      IF it = 'IMAGE' THEN DO      RENDER_TYPE
RenderStr      IF pal = 1 THEN RENDER_TYPE 'CUST'      EXECUTE
```

ARexx w praktyce - część 4

Cholok

(c) Polski Portal Amigowy (www.ppa.pl)

```
IF (RC ~= 0) THEN DO                                ErrorText = 'Render Error'                                CALL LabelErr
    END                                                END
```

Ten fragment zajmuje się renderowaniem do pożądanego trybu. Przy trybach RAW jest on całkowicie zbędny.

```
IF FileNr = 1 THEN DO                                pad = 0                                GETNUMBER 'Enter Pad Space' 0 0 999
    IF (RC = 0) THEN pad = ADPRO_RESULT                END
```

Tutaj możemy opcjonalnie ustawić wielkość dopełnienia. Nie jest to potrzebne, ale dla przykładu jest. Lepiej jest stosować wielkość parzystą.

```
/* save cdxl frame, noaudio, nopad */                SAVER CDXL DQ || XLFile || DQ it FRAMENUM FileNr PADSPACE
pad SAVEPAD ivd    IF (RC ~= 0) THEN DO                ErrorText = 'Save Error'                CALL LabelErr
END
```

Komenda SAVER zapisuje obraz w pożądanym formacie. Zmienna it i ivd użyte są jako parametry.

```
LabelErr: /* close file */ SAVER CDXL DQ || XLFile || DQ QUIT RDEPTH UNLOCKED PSTATUS UNLOCKED
CLOSE_RENDER_SCREEN CLEAR_RAW CLEAR_RENDERED RELEASE_ADPRO IF (EXISTS( TempDefaults ))
THEN DO                                LOAD_DEFAULTS TempDefaults                                IF (RC ~= 0) THEN OKAY1 'Error restoring
settings.'                                ADDRESS COMMAND 'Delete &NUL: TempDefaults END OKAY1 ErrorText
EXIT 0
```

Tutaj kończymy skrypt, jednocześnie wszystkie skoki w czasie błędów lądują tutaj. Tutaj SAVER CDXL zamyka plik, nic się nie dzieje jeżeli plik nie istnieje. Reszta kasuje bufor, zwalnia Adpro i przywraca zapisane ustawienia. Komenda OKAY1 wyświetla komunikat błędu lub tekst kończący konwersję. EXIT zwraca tzw. return code, podobnie jak programy w CLI.

Co do działania skryptu. Adpro jest programem w miarę szybkim, ale przy każdej operacji wyświetla okienko z paskiem postępu. To bardzo zwalnia całość konwersji. Druga rzecz to problem palety. W plikach CDXL zawsze jest 12-bitowa co rodzi szereg problemów. Adpro generuje zawsze 24-bitową, więc to wyświetlone nie równa się to co zapisane. Jedynie HAM6 jest wolny od tego (prawdopodobnie EHB też). Do 32 kolorów nie będzie to problem. bo można użyć innych programów, ale powyżej będzie to problem. Przy HAM8 nie widać tego tak bardzo, ale zawsze można ustawić własną, stałą paletę. Generator HAM8 radzi sobie dobrze z dowolną paletą. Oczywiście, skrypt generuje filmik bez dźwięku. Należy go dodać osobno programem xlaudio. Gwoli ścisłości odtwarzacz cdgsxl nie radzi sobie dobrze z filmami bez dźwięku. Bez względu na parametr xlspeed odgrywa je zawsze z maksymalną prędkością.