jąc im poprawną współpracę. Dlatego modem najbezpieczniej jest połączyć z komputerem, gdy jest on odłączony od zasilania (ale uziemiony!).

Powróćmy do łączenia modemu i komputera. Służy do tego, jak już wspomniałem, gruby przewód, zakończony z obu stron dość dużymi, płaskimi wtyczkami o 25 bolcach (z jednej strony) i 25 dziurach (z drugiej). Te wtyczki nazywane są fachowo DIP-25 typu męskiego i żeńskiego. Wtyk „męski" podłączamy do modemu, natomiast „żeński" do komputera, do gniazda oznaczonego słuchawką telefoniczną.

Na wtyczkach podłączanych do modemu i komputera bardzo często znajdują się śrubki lub pokrętła, zakończone gwintem. Dzięki nim można przykręcić wtyczkę do obudowy (komputera, modemu), zapewniając trwałe i bezpieczne - dla użytkowanego sprzętu - połączenie. Następnie należy włączyć zasilacz modemu i podłączyć go do „bzykacza". Nie będę zgadywał, w jaki sposób to ma nastąpić, czy jedynie poprzez podłączenie go do sieci, czy też w inny sposób - jakimś ukrytym wyłącznikiem. Niemniej, jeśli modem i komputer są już połączone, włączenie modemu objawi się zapaleniem diod: TR, MR, PW - na płycie czołowej modemu, informując o poprawnym połączeniu wszystkich elementów. Wyłączamy modem. Teraz można podłączyć linię telefoniczną do modemu (przewodem zakończonym charakterystyczną, kwadratową wtyczką - nawiasem mówiąc, standardową na Zachodzie), a następnie włączyć modem i (później) komputer. Jeśli nie odleciał lub nie zaczął wydawać podejrzanych odgłosów wraz z niespotykanymi na co dzień efektami wizualno-zapachowymi - to oznacza, że można przystąpić do... bzykania.

Na dzisiaj... już koniec. W następnym odcinku opiszę, w jaki sposób można wykonać pierwsze połączenie oraz czym je wykonać. Podam też najważniejsze polecenia modemu.

CZUJ DRUT

wasz SysOp „Mischer"

PS.1. Na razie ATMA BBS jest w fazie donoszenia. Jeśli nastąpi szczęśliwe rozwiązanie, dam wam znać. Trzymajcie kciuki.

PS.2. Tradycyjne: CDNPN (Ciąg Dalszy Na Pewno Nastąpi)

Rys.3 - jak połączyć komputer z modemem?

NIE BÓJ SIĘ BITPLANÓW

czyli obrazek oczami komputerów

Tomasz Gnyp

Na pewno, siedząc czasem przed komputerem i patrząc na jakiś ciekawy obrazek na ekranie monitora, zastanawiasz się, skąd komputer „wie", że dany punkt ma mieć taki a nie inny kolor, dlaczego np. w jakimś trybie graficznym określona jest dana liczba punktów na ekranie oraz jakaś (maksymalna) liczba kolorów, taka... a nie zupełnie inna. W tym artykule postaram się wyjaśnić Tobie, Drogi Czytelniku, w jaki sposób zapisane są w pamięci komputera informacje o wyświetlanym obrazie. Fakt, iż artykuł znajduje się w rubryce poświęconej komputerom Atari ST/Falcon spowodowany jest tym, że przykłady trybów graficznych, jakie w nim przedstawię pochodzą z tych właśnie komputerów. Ogólna zasada zapisu obrazu i jego kolorów jest jednak podobna dla wielu innych komputerów - np. Amiga, IBM...

Na początku chciałbym wyjaśnić kilka terminów, których będę używał w dalszej części artykułu, przy opisie struktury obrazu i jego kolorów:

- **pamięć obrazu** (*Video RAM*) - jest to określony obszar w pamięci RAM komputera (czyli tej, do której można zapisywać dane), zawierający informacje o aktualnie wyświetlanym obrazie. Adres (numer komórki pamięci), od którego zaczyna się *Video RAM*, zapisany jest w rejestrach układu, zajmującego się przetwarzaniem danych z pamięci na sygnał doprowadzany do gniazda monitorowego. Rozmiar *Video RAM* zależy od aktualnej rozdzielczości (liczby linii na ekranie i liczby punktów w każdej z nich) obrazu i liczby dostępnych na nim jednocześnie kolorów.
- **tryb graficzny** - sposób wyświetlania na ekranie monitora obrazu o określonej rozdzielczości i maksymalnej liczbie kolorów.
- **pióra** - czyli zestaw kolorów dostępnych jednocześnie na ekranie, w danym trybie graficznym. Jedno pióro możemy „napełnić" różnymi kolorami z dostępnej palety. Jeżeli np. piórem o numerze 2 i kolorze czerwonym narysujemy na ekranie kreskę i potem zmienimy kolor tego pióra na niebieski, to kolor narysowanej wcześniej kreski RÓWNIEŻ się zmieni!

Na przykład w programie graficznym „DEGAS" symbolami piór są kratki, wypełnione ustalonymi kolorami, wyświetlane w górnej części ekranu, pod listwą menu.

- **rejestry kolorów** – są odpowiednikiem piór w... pamięci komputera. Rejestry te należą do układu wizyjnego i zawierają kody RGB kolorów, którymi wypełnione są pióra.
- **paleta kolorów** – wszystkie kolory, jakie może wygenerować układ wizyjny w danym komputerze. W **Atari ST** paleta kolorów obejmuje 512 barw, w **Atari STE** – 4096 barw, w **Falconie** – 262144. Przykład? Proszę: paleta może obejmować 4096 kolorów, a na ekranie jednocześnie można użyć tylko czterech różnych piór (w średniej rozdzielczości **Atari STE**).
- **kod RGB** – każdy kolor daje się przedstawić za pomocą kombinacji odpowiedniego nasycenia trzech podstawowych barw: czerwonej (*Red*), zielonej (*Green*) i niebieskiej (*Blue*). Im więcej możliwych poziomów nasycenia RGB, tym większa jest paleta kolorów.
- **bajt** – podstawowa jednostka określająca pojemność pamięci lub rozmiar danych. Jeden bajt składa się z ośmiu bitów.
- **Słowo** – dwa sąsiednie bajty w pamięci komputera. Słowo można podzielić również na 16 bitów, ponumerowanych od 0 do 15, przy czym numeracja rozpoczyna się od prawej strony i rośnie w kierunku lewej, czyli poszczególne numery pozycji binarnej określane są następująco:

  15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

  a przykładowe wartości bitów jakiegoś słowa:

  1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0.

  Bit może przyjmować wartość 0 – zgaszony, lub 1 – zapalony (ustawiony). W celu obliczenia dziesiętnej równowartości liczby binarnej, należy pomnożyć kolejne wartości bitów (0 lub 1) przez liczbę dwa, podniesioną do potęgi określonej numerem pozycji, w jakiej znajduje się dany bit, oraz dodać do siebie wszystkie 16 wyników mnożenia. W naszym przypadku:

  1*2^15 + 0*2^14 + 0*2^13 + 1*2^12 + 1*2^11 +...+ 1*2^2 + 0*2^1 + 0*2^0 = 39028
- **bitplany** – Coś takiego, jak bitplany, nie istnieje fizycznie w pamięci komputera, ponieważ jest ona jednowymiarowa (jest to po prostu szereg komórek). Pojęcie to wprowadzamy, aby zrozumieć zasadę przyporządkowywania odpowiedniego pióra – – danemu punktowi.

CO TO SĄ BITPLANY?

Przyjrzyjmy się rysunkowi. Przedstawia on organizację dwóch pierwszych linii ekranu, w poszczególnych trybach graficznych, w pamięci *Video RAM*. Pierwsza część rysunku odpowiada niskiej rozdzielczości **Atari ST/STE**. Jedna linia w tej rozdzielczości obejmuje 160 bajtów, ponumerowanych od 0 do 159, lub 80 słów (jedno słowo = dwa bajty). Prostokąty na rysunku, umieszczone jeden za drugim (w głąb kartki), symbolizują kolejne słowa w *Video RAM*. Prostokąt o numerze 0 oznacza słowo o numerze 0, czyli dwa bajty o numerach 0 i 1. Następny prostokąt to słowo o numerze 2, czyli dwa bajty o numerach 2 i 3 itd...

Dlaczego jednak przedstawiłem je w taki sposób, a nie w dwóch wymiarach, skoro posiadają kolejne numery i w pamięci komputera ułożone są „jedno za drugim"? Właśnie te narysowane „warstwy" pomagają zrozumieć pojęcie bitplanów w komputerach **Atari ST/STE**. W niskiej rozdzielczości mamy więc cztery „warstwy", czyli cztery bitplany. Do pierwszego bitplanu należą słowa o numerach 0, 8, 16, 24,... czyli, jeżeli przez „n" oznaczymy początek *Video RAM*, to pierwszy bitplan składa się ze słów „n + k*8", gdzie k = 0, 1, 2, 3,..., natomiast liczba 8 oznacza cztery słowa po dwa bajty na słowo.

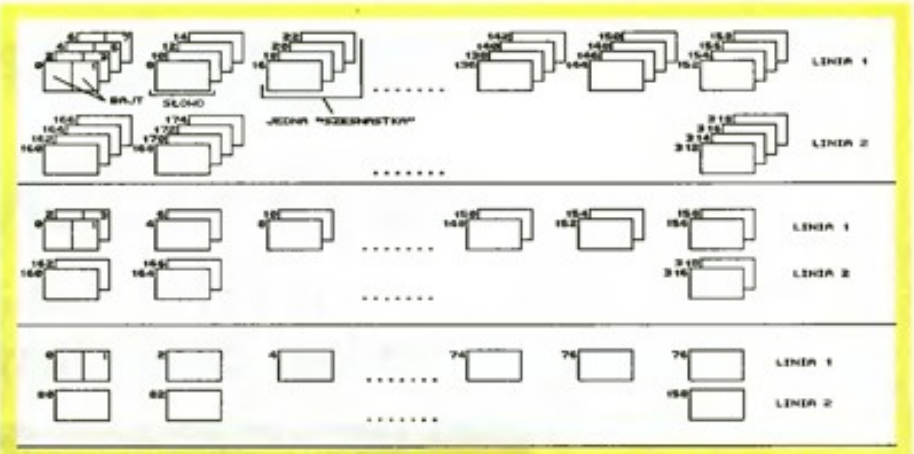
Oczywiście, bitplany w kolejnych liniach wyglądają tak samo, np. pierwsze słowo pierwszego bitplanu w drugiej linii to n + 20*8, w trzeciej = n + 40*8, czwartej = n + 50*8 itd... Widzimy więc,

```
'
' Program 1  GFA-BASIC v. 3.50 F (c) 1993  Tomasz Gnyp
' Nie bój się bitplanów
'
IF XBIOS(4)<>0
  PRINT "Proszę uruchomić program w niskiej rozdzielczości"
  END
ENDIF
'
CLS                              ! czyść ekran
n%=XBIOS(2)                      ! adres początkowy Video RAM
'
slowo1%=&X1111111111100111       ! ustalamy wartość słowa
k%=0                             ! pierwsze słowo od pocz. linii
FOR linia%=0 TO 199              ! 200 linii
  adres%=n%+k%*8                 ! obliczenie adresu danego słowa w
  '                                pierwszym bitplanie
  ADD adres%,linia%*160          ! słowa mają być wyświetlane w
  '                                pionowym pasie, więc aby następne
  '                                znalazło się pod poprzednim,
  '                                musi być o 160 bajtów dalej
  DPOKE adres%,slowo1%           ! wpisanie pod obliczony adres
  '                                ustalonej wartości
NEXT linia%                      ! następna linia
'
slowo1%=&X1111011111101111
slowo2%=&X1111011111101111
k%=2                             ! trzecie słowo od początku linii
FOR linia%=0 TO 199
  adres%=n%+k%*8                 ! pierwszy bitplan
  ADD adres%,linia%*160
  DPOKE adres%,slowo1%
  '
  adres%=n%+2+k%*8               ! drugi bitplan
  ADD adres%,linia%*160
  DPOKE adres%,slowo2%
NEXT linia%
'
slowo1%=&X1100111011110011
slowo2%=&X1100111011110011
k%=4                             ! piąte słowo od początku linii
FOR linia%=0 TO 199
  adres%=n%+4+k%*8               ! trzeci bitplan
  ADD adres%,linia%*160
  DPOKE adres%,slowo1%
  '
  adres%=n%+6+k%*8               ! czwarty bitplan
  ADD adres%,linia%*160
  DPOKE adres%,slowo2%
NEXT linia%
'
slowo1%=&X1100111001111011
slowo2%=&X1111010101011001
slowo3%=&X1101101011010010
slowo4%=&X1110111011100011
k%=6                             ! siódme słowo od początku linii
FOR linia%=0 TO 199
  adres%=n%+k%*8                 ! pierwszy bitplan
  ADD adres%,linia%*160
  DPOKE adres%,slowo1%
  '                                do obliczonego adresu pierwszego
  '                                bitplanu wystarczy dodawać kolejno
  '                                liczbę dwa, aby uzyskać
  '                                adresy następnych bitplanów
  '
  DPOKE adres%+2,slowo2%         ! drugi bitplan
  DPOKE adres%+4,slowo3%         ! trzeci bitplan
  DPOKE adres%+6,slowo4%         ! czwarty bitplan
NEXT linia%
END
```

że do jednego bitplanu każdej linii należy po dwadzieścia „co czwartych" słów. Podobnie sytuacja wygląda z drugim bitplanem, tylko, że nasz wzór na obliczanie kolejnych słów przyjmuje teraz nieco inną postać: „n+2 + k*8" (n+2, ponieważ pierwsze słowo *Video RAM*, należące do drugiego bitplanu, ma numer 2). Trzeci bitplan, to „n+4 + k*8", a czwarty, to „n+6 + k*8".

Zapamiętajmy więc, że w niskiej rozdzielczości obraz składa się z:

- 200 linii, każda po 320 punktów
- 4 bitplanów
- 20 słów przypadających na każdy bitplan w jednej linii
- 80 słów, czyli 160 bajtów przypadających na każdą linię
- cały obraz zajmuje 200 linii * 4 bitplany * 20 słów na bitplan * 2 bajty na słowo = 32000 bajty
- adres (czyli numer) danego słowa, należącego do pierwszego bitplanu, obliczamy ze wzoru: „n + k*8", gdzie „n" jest adresem początku *Video RAM*, a „k" numerem słowa, dla drugiego bitplanu: „n+2 + k*8", dla trzeciego: „n+4 + k*8" i dla czwartego bitplanu: „n+6 + k*8".



Rysunek – Odwzorowanie w Video RAM dwóch pierwszych linii ekranu, w trzech dostępnych trybach graficznych

W średniej rozdzielczości (druga część rysunku) linia składa się z dwóch bitplanów, przy czym do każdego bitplanu należy po 40 słów o numerach: „n + k*4", lub „n+2 + k*4", gdzie liczba 4 oznacza dwa słowa po dwa bajty na słowo. Zapamiętajmy, że w średniej rozdzielczości obraz składa się z:

- 200 linii, każda po 640 (!) punktów
- 2 bitplanów
- 40 słów przypadających na każdy bitplan w jednej linii
- 80 słów, czyli 160 bajtów przypadających na każdą linię
- cały obraz zajmuje 200 linii * 2 bitplany * 40 słów na bitplan * 2 bajty na słowo = 32000 bajty
- adres danego słowa należącego do pierwszego bitplanu obliczamy ze wzoru: „n + k*4", gdzie n jest adresem początku *Video RAM*, a k numerem słowa, a dla drugiego bitplanu: „n+2 + k*4".

Trzecia część rysunku przedstawia budowę linii obrazu w wysokiej rozdzielczości. Tutaj jest ona najprostsza, ponieważ mamy tylko jeden bitplan i do niego należą po prostu kolejne słowa *Video RAM*, czyli „n + k*2", gdzie cyfra 2 symbolizuje jedno słowo, czyli dwa bajty. Zapamiętajmy informację o obrazie w wysokiej rozdzielczości:

- 400 linii (!), każda po 640 punktów
- 1 bitplan
- 80 słów, czyli 160 bajtów przypadających na każdą linię
- cały obraz zajmuje 400 linii * 1 bitplan * 80 słów na bitplan * 2 bajty na słowo = 32000 bajty
- adres danego słowa należącego do jednego bitplanu obliczamy ze wzoru: „n + k*2", gdzie n jest adresem początku *Video RAM*, a k numerem słowa.

Jak widzimy, w zależności od liczby bitplanów w danym trybie graficznym zmienia się rozdzielczość pionowa lub pozioma obrazu. Od liczby bitplanów zależy również liczba piór, możliwych do użycia w danym trybie (o tym jednak w dalszej części artykułu). Teraz proponuję wpisanie zamieszczonego obok listingu, który napisany został w *GFA Basic-u* (wersja 3.0) i poprawnie pracuje w trybie niskiej rozdzielczości **Atari ST/STE**. Program wykorzystując podane przeze mnie informacje o bitplanach, wyświetla na ekranie cztery pionowe paski, używając kolejno jednego bitplanu, potem dwóch pierwszych i dwóch ostatnich oraz... wszystkich bitplanów. Zwróć uwagę na zależność szerokości paska i jego struktury, od liczby i układu zapalanych bitów w poszczególnych słowach.

Po uruchomieniu programu widać, że na ekranie pojawiły się cztery paski o różnej strukturze i różnych kolorach. Jeżeli zmienimy wartości zmiennych „**slowo1%**"..."**slowo4%**", zamieniając dowolnie zera z jedynkami (i odwrotnie), otrzymamy paski o różnych strukturach i różnych kolorach. Łatwo można zauważyć związek pomiędzy układem zer i jedynek w wartościach słów wpisywanych do kolejnych bitplanów, a różnorodnością i ilością pojawiających się kolorów w obrębie jednego paska. Ponadto widać, że jedna pozycja binarna (czyli jeden bit) w danym słowie odpowiada jednemu punktowi na ekranie. Jak w takim razie wygląda ta zależność wartości słowa i koloru punktu na ekranie? Tu właśnie dochodzimy do „sedna" bitplanów, czyli do tego, jak wygląda interpretacja koloru punktu w pamięci komputera, o czym dokładniej już za miesiąc. ◄

NIE BÓJ SIĘ BITPLANÓW

czyli obrazek oczami komputera (część II)

Tomasz Gnyp

Miesiąc temu wyjaśniłem wam podstawowe pojęcia, związane z obrazem wyświetlanym przez komputer. Zacząłem tłumaczyć, jak reprezentowany jest ekran w pamięci komputera. Uruchomiliśmy też nasz pierwszy program, oparty o tzw. bitplany. Obiecałem wówczas dokładniej przedstawić, na czym one „polegają".

Na początek zajmiemy się czymś, co można określić jako...

INTERPRETACJA KOLORU PUNKTU W PAMIĘCI KOMPUTERA

Spójrzmy na rysunek 1. Pięć kratek w górnej części rysunku przedstawia pięć pierwszych (licząc od lewego, górnego rogu) punktów ekranu. Punkty ponumerowane są od zera do czterech i załóżmy, że mają następujące kolory: niebieski, błękitny, żółty, fioletowy i czerwony. Po środku rysunku mamy zestaw szesnastu piór (ponumerowanych od zera do piętnastu), o ustalonych już wcześniej przykładowych, wybranych z palety kolorach. Pod każdym prostokątem, symbolizującym jedno pióro, znajduje się kod RGB koloru tego pióra. Te właśnie kody zapisane są w kolejnych rejestrach kolorów układu wizyjnego Atari ST/STE/Falcon. Na dole rysunku cztery prostokąty, podzielone na szesnaście kratek-bitów, odpowiadają czterem pierwszym słowom Video RAM, o numerach 0, 2, 4 i 6 (pamiętamy, że słowa numerujemy co dwa, ponieważ są to dwa bajty), należącym do czterech kolejnych bitplanów. Kratki wypełnione kolorem czarnym symbolizują bity zapalone - o wartości 1, a kratki wypełnione kolorem białym - bity zgaszone - o wartości zero. Pomiędzy piórami a słowami z Video RAM znajdują się kombinacje bitów z poszczególnych bitplanów, oznaczające numer użytego pióra do narysowania danego punktu na ekranie. Prześledźmy razem proces kodowania kolorów punktów na ekranie za pomocą bitplanów.

Zastanówmy się nad pierwszym punktem na ekranie, o numerze zero i kolorze niebieskim. Popatrzmy teraz, jak wyglądają pierwsze od lewej (o numerach 15) bity kolejnych czterech słów Video RAM, należących do kolejnych czterech bitplanów. Rozpocznijmy od słowa oznaczonego numerem 6, należącego do czwartego bitplanu. Bit o numerze 15 w tym słowie ma wartość 0, bit 15 w słowie 4 ma wartość 1, bit 15 w słowie 2 ma wartość 0, a bit 15 w słowie 0 ma wartość 1. Zestawmy ze sobą te cztery bity, przypominam, począwszy od czwartego bitplanu, czyli słowa o numerze 6. Otrzymamy wartość binarną 0101, czyli przeliczając na system dziesiętny: 0101 = 0*23 + 1*22 + 0*21 + 1*20 = 5

Obliczona wartość z zestawienia rozpatrywanych bitów nr 15. kolejnych słów, należących do czterech bitplanów - daje numer pióra, którym narysowany jest na ekranie odpowiedni punkt (w naszym przypadku pierwszy w lewym, górnym rogu ekranu). To pióro, o numerze 5, przy takich wartościach rozpatrywanych bitów nr 15, jest ściśle związane z tym punktem, tzn. punkt będzie miał taki kolor, jakim „wypełnione" jest pióro 5. Na rys. 1 owe pióro ma kolor niebieski, zatem punkt o numerze 0 ma również kolor niebieski. Jeżeli zmienimy kolor pióra, to również zmieni się kolor punktu 0. Związanie punktu 0 z innym piórem, wymagałoby zmiany wartości bitów 15 ze słów 0. 2. 4, 6.

Punkt o numerze 1 ma kolor błękitny, popatrzmy na bity o numerach 14 naszych czterech słów:

bit 14, słowo 6 - 0
bit 14, słowo 4 - 0
bit 14, słowo 2 - 1
bit 14, słowo 0 - 0,

czyli binarnie 0010, dziesiętnie 2.

Pióro o numerze 2 jest „wypełnione" kolorem błękitnym. Punkt o numerze 1 (drugi od początku ekranu), któremu odpowiadają bity o numerach 14 (drugie od lewej kolejnych słów, należących do kolejnych bitplanów), związany jest z piórem o numerze 2, co wynika z kodu binarnego zapisanego w powyższych bitach, stąd punkt ten ma również kolor błękitny.

Spróbujmy teraz przeprowadzić odwrotną analizę. Zastanówmy się np. nad czwartym punktem od początku ekranu, o numerze 3. Punktowi temu odpowiadać będą bity o numerach 12 z czterech bitplanów. Poszukajmy teraz pióra „wypełnionego" takim kolorem, jak punkt 3 - czyli fioletowym. Pióro o tym kolorze posiada numer 7, binarnie 0111. Czyli licząc od ostatniego bitplanu - słowa o numerze 6, bity 12 powinny mieć kolejno wartości 0, 1, 1, 1. Sprawdźmy, czy zgadza się to z rysunkiem. Zgadza się.

W podobny sposób, jak robiliśmy to dla punktu o numerze 3, określa się kolory wszystkich punktów na ekranie. Najpierw zastanawiamy się, jakiego koloru ma być nasz punkt, potem „wypełniamy" takim kolorem wybrane pióro, czyli wpisujemy odpowiedni kod RGB do odpowiedniego rejestru koloru. Następnie odnajdujemy w Video RAM cztery bity odpowiadające naszemu punktowi i ustalamy ich wartości, zgodnie z numerem wybranego pióra. W taki właśnie sposób działają wszystkie programy graficzne. Zastanówmy się jeszcze...

W JAKI SPOSÓB ODNALEŹĆ W VIDEO RAM BITY ODPOWIADAJĄCE DANEMU PUNKTOWI?

Z naszych wcześniejszych rozważań wynika, że szesnaście kolejnych czwórek bitów, o numerach od 15 do 0, należących do czterech kolejnych słów, odpowiada kolejnym (od 0 do 15) szesnastu punktom na ekranie. A co z dalszymi punktami? Otóż punkt o numerze 16 (1 * 16 + 0), odpowiada pierwszej czwórce bitów (nr 15) ze słów 8, 10, 12, 14 (patrz rysunek 1), punkt 17 (1 * 16 + 1) - drugiej czwórce bitów (nr 14) słów 8, 10, 12, 14 itd. Punkt o numerze 32 (2 * 16 + 0) odpowiada pierwszej czwórce bitów (nr 15) ze słów 16, 18, 20, 22, a punkt np. 36 (2 * 16 + 4) odpowiada piątej czwórce bitów (nr 11) ze słów 16, 18, 20, 22.

Widać więc, że w celu odnalezienia odpowiednich słów musimy najpierw określić, w której „szesnastce" czwórek bitów (jedna „szesnastka" odpowiada czterem słowom, każde po 16 bitów) znajduje się nasz punkt na ekranie, a potem jaki numer bitu w tej szesnastce mu odpowiada. Ograniczmy na razie nasze rozważania do pierwszej linii ekranu. Weźmy np. punkt o współrzędnej x=179. Wykonujemy następujące obliczenia:

179: 16 = 11 i reszta z dzielenia 3

Całkowita część wyniku określa nam „szesnastkę", natomiast reszta mówi o tym, która czwórka bitów, (licząc od lewej, czyli od numeru 15) odpowiada naszemu punktowi. Wykonajmy więc obliczenie, które da nam numer pierwszego słowa z czterech stanowiących tę jedenastą „szesnastkę":

11 * 8 (osiem bajtów przypadających na cztery słowa) = 88

Wyliczyliśmy więc, że nasze „jedenaste" cztery słowa rozpoczynają się od numeru 88 i mają kolejne numery 88, 90, 92, 94. Reszta,

określająca bit, wynosiła 3, czyli będą interesowały nas cztery bity o numerach 13 (liczymy od bitu numer 15!), należące do słów 88, 90, 92, 94.

Jeżeli teraz nasz punkt o współrzędnej x=179, znajdowałby się w linii ekranu o numerze 10 (czyli jedenastej linii, bo numerujemy od zera!), to do obliczonych numerów słów 88, 90, 92, 94 należy dodać ilość wszystkich opuszczonych „szesnastek". Ponieważ na jedną „szesnastkę" przypadają cztery słowa, czyli osiem bajtów, a linia ma 160 bajtów, więc opuszczając 10 linii (nasz punkt znajduje się w jedenastej) musimy dodać 10 * 160, czyli 1600 bajtów. Ostatecznie mamy:

- punkt o współrzędnych x=179, y=10 (x i y liczone są, oczywiście, od zera)
- 179: 16 = 11, r. 3 - 11-ta „szesnastka", trzeci bit od lewej, czyli nr 13,
- 11 * 8 = 88 - na każdą „szesnastkę" przypada osiem bajtów.
- 88 + 10 * 160 = 1688 - opuszczamy 10 linii, czyli 10 razy po 160 bajtów,

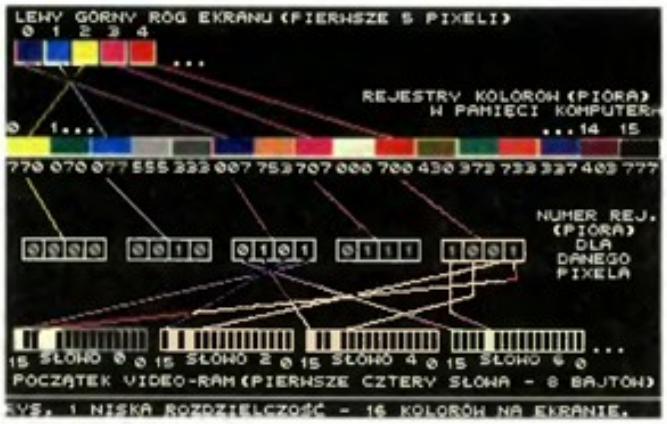
zatem interesują nas cztery bity o numerach 13, należące do czterech kolejnych słów o numerach 1688, 1690, 1692, 1694, licząc od początku Video RAM. Aby teraz obliczyć właściwy adres tych słów w pamięci komputera, należy do numerów tych słów dodać adres początku Video RAM.

Przedstawiony obok program umożliwia Ci wybranie dowolnego punktu na ekranie, a następnie odnajduje on odpowiednie cztery bity w Video RAM i ustala ich wartości tak, aby wskazywały odpowiednie pióro.

Jak widać, wszystkie obliczenia wykonywane przez program są dokładnie takie same jak nasze. Na początku programu można również zmienić kod używanego pióra na inny i sprawdzić, jaki ma to wpływ na kolor punktu.

Do tego artykułu dołączone są jeszcze dwa rysunki - nr 2, przedstawiający zasadę kodowania kolorów za pomocą bitplanów w średniej rozdzielczości Atari ST/STE i w wysokiej rozdzielczości - rysunek nr 3. W średniej rozdzielczości kodowanie to odbywa się podobnie, jak w niskiej, z tym, że mamy do dyspozycji tylko dwa bitplany, czyli po dwa bity przypadające na jeden punkt na ekranie. Za pomocą dwóch bitów można uzyskać następujące kombinacje binarne: 00=0, 01=1, 10=2, 11=3, czyli każdy punkt na ekranie możemy „związać" tylko z czterema różnymi piórami, o numerach 0,1,2,3. Dzięki temu ograniczeniu możemy jednak uzyskać dwa razy więcej punktów w linii, niż w niskiej rozdzielczości. Dzieje się tak, ponieważ na każde 16 punktów przypadają 4 bajty (dwa bitplany), a nie 8 (cztery bitplany), a wiedząc, że jedna linia to 160 bajtów, mamy:

- rozdz. niska: 160: 8 = 20 „szesnastek", czyli 20 * 16 = 320 punktów
- rozdz. średnia: 160: 4 = 40 „szesnastek", czyli 40 * 16 = 640 punktów



W wysokiej rozdzielczości zasada kodowania kolorów punktów jest najprostsza. Mamy jeden bitplan, czyli 1 bit przypadający na 1 punkt. Jeżeli bit jest zapalony, czyli równy 1, odpowiadający mu punkt na ekranie ma kolor pióra o numerze 1. Jeśli bit jest zgaszony, równy 0, to odpowiedni punkt ma kolor pióra o numerze 0. W tej rozdzielczości ograniczenie piór (a zatem i różnych kolorów, jakie można jednocześnie uzyskać na ekranie) do dwóch daje nam zwiększenie liczby linii na ekranie. Wiemy, że na każde szesnaście punktów przypadają dwa bajty (jeden bitplan), na jedną linię przypada 80 bajtów, a cały obraz zajmuje 32000 bajty, policzmy:

- rozdz. wysoka: 80: 2 = 40 „szesnastek", czyli 40 * 16 = 640 punktów

32000: 80 = 400 linii.

Jeżeli chcielibyśmy zwiększyć ilość możliwych jednocześnie do uzyskania kolorów na ekranie, czyli ilość dostępnych piór, trzeba zwiększyć ilość bitplanów. Oczywiście do takiej zmiany zapisu obrazu w pamięci musi być również przystosowany układ wizyjny komputera. W modelach ST/STE nie ma niestety możliwości uzyskania w „normalny" sposób, czyli bez stosowania sztuczek programowych, innych rozdzielczości niż trzy standardowe. Zwiększenie możliwej do uzyskania ilości kolorów wykonuje się poprzez bardzo szybką zmianę kolorów w paletach w ciągu wyświetlania jednej linii ekranu (64 mikrosekundy!). Ten sposób wprowadza jednak pewne ograniczenia i zajmuje dość dużo czasu procesorowi komputera. Uzyskanie większej ilości punktów w linii, lub większej ilości linii na ekranie poprzez pozbycie się ramki (tzw. noborder) polega na pewnego rodzaju „zakłóceniu" pracy układu wizyjnego. I ten sposób powoduje duże obciążenie procesora i wymaga wysokiego zaawansowania w programowaniu w języku maszynowym oraz bardzo dokładnego zsynchronizowania programu procesora z pracą układu wizyjnego.

W nowszych modelach **Atari**, mianowicie **TT** i **Falcon**, zastosowano nowocześniejsze układy wizyjne. Np. w **Falconie** znajduje się wysokiej klasy procesor wizyjny, czyli układ, który może być programowany przez użytkownika i przy jego pomocy (układu), bez udziału procesora głównego można uzyskać na ekranie rozdzielczość do ok. 1200 punktów w linii przy ok. 1000 linii i 65536 jednocześnie wyświetlanych kolorach na ekranie. Tak duża, w porównaniu z modelami **ST/STE**, liczba dostępnych naraz kolorów możliwa jest dzięki zastosowaniu 16 bitplanów. Inny tryb graficzny **Falcona**, zgodny ze standardem VGA, pozwala na uzyskanie 256 różnych kolorów na ekranie. Taką liczbę kombinacji można uzyskać za pomocą 8 bitplanów.

Jak widać, „przesiadając się" na lepszy komputer nie możemy uchronić się od „towarzystwa" bitplanów, jest ich wręcz więcej, jednak zasada ich wykorzystania jest zwykle podobna lub nawet taka sama, jak ta opisana przeze mnie w tym artykule. Opanowanie sposobu posługiwania się bitplanami daje programiście możliwość uzyskiwania np. efektów znajdowania się pewnych obiektów graficznych pod innymi, czego doskonałym przykładem może być scroll (przesuwający się tekst) w dolnej części ekranu menu starego, dobrze chyba wszystkim znanego Union Demo. Pod przesuwającymi się literami w tym właśnie scrollu znajduje się górski krajobraz. Sposób, w jaki można uzyskać taki efekt, przy użyciu bitplanów, jest już jednak tematem na odrębny artykuł. ◄