ATARI - COMMODORE - ZX SPECTRUM - AMSTRAD CPC - AMIGA - APPLE - ACORN - IBM - NINTENDO

RETROKOMP

Mikrokomputery wczoraj i dziś

15

SILLY VENTURE 2019

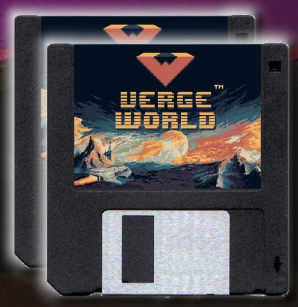
Keeping The Legacy Of The Atari Scene Alive!

VERGE WORLD™

ISSN 2450-5862

WYDANIE SPECJALNE Z OKAZJI SILLY VENTURE 2019:

Demo gry Verge World (Amiga/Atari ST)

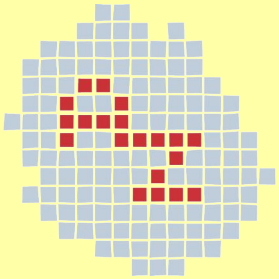


RetroKomp 15

WYDANIE SPECJALNE - TYLKO O ATARI

WIĘCEJ WĘGLA

Zapraszamy do słuchania podcastu Adama Zalepy pod tytułem "Więcej Węgla". Ostatnio ukazał się jubileuszowy 10-ty odcinek z rozmową z Piotrem "Piterem" Krużyckim, którego przedstawiać nie trzeba. Panowie mówią przede wszystkim o parametrach technicznych komputerów Amiga 1200 oraz Atari Falcon.



Odcinek trwa 30 minut i można go pobierać jako plik MP3 lub słuchać pod poniższym adresem:

https://anchor.fm/amiga-net-pl

VERGE WORLD

Powstaje zupełnie nowa gra na Atari ST! Demonstracyjną wersję możecie uruchomić z dyskietki załączonej do naszego magazynu. Pamiętajcie, że jest to pierwsze demo, a gra jest ciągle rozwijana przez grupę Retro Bones. Dlatego w pełnej wersji pojawi się z pewnością dużo więcej ciekawych elementów.

Verge World ukaże się równolegle na Amigę, a planowany termin wydania to rok 2020. Więcej informacji można przeczytać tutaj:

https://www.facebook.com/RetroBones-Games/

Dwie dwudziestki

ADAM ZALEPA

W tym roku zdecydowałem się na podróż do Gdańska na imprezę Silly Venture, tak więc niedługo się zobaczymy. Bardzo się z tego cieszę, bo do tej pory imprezy Atarowców znam jedynie z opowieści i relacji znajomych. Świat Atari jest na tyle interesujący, że kupiłem nawet swój własny komputer spod znaku góry Fuji. To nic wielkiego, Atari 520 ST z 4 MB pamięci, ale dla mnie to krok w zupełnie nowym kierunku.

Ten numer RetroKompa jest specjalnym wydaniem — tylko o Atari. Jak widzicie, jest nieco mniejszy objętościowo, ale za to dołączamy do niego demo zupełnie nowej gry. Będzie ona wydana w dwóch wersjach — na Amigę oraz Atari ST. W tej chwili możecie zapoznać się z grywalną wersją demonstracyjną i mam nadzieję, że przypadnie ona Wam do gustu. Nie da się ukryć, że Verge World do komfortowej rozgrywki wymaga nieco mocniejszego komputera niż zwykły 7-megahercowy procesor, ale uwierzcie mi — warto zagrać. Koledzy z grupy Retro Bones starają się uzyskać jak największą optymalizację algorytmów, dlatego gra uruchamia się praktycznie na każdym sprzęcie, natomiast pewnych rzeczy się nie przeskoczy. Procesor 020 lub 030 daje już odpowiednią wydajność, więc uważam, że jest to całkiem niezły kompromis.

Jeśli jeszcze nie znacie naszego magazynu, dodam, że ukazuje on się od 2015 roku. W tym roku wydaliśmy także 2 anglojęzyczne numery, które zostały bardzo dobrze przyjęte za granicami naszego kraju. W październiku prezentowaliśmy go na imprezie Amiga34 w Niemczech, zebraliśmy sporo uwag i opinii, które przełożą się również na zmiany w polskiej edycji. Chcemy dalej pisać o użytkowym charakterze komputerów retro, choć oczywiście nie pomijamy gier — to byłoby niemądre. Pisaliśmy o świetnych tytułach na różne modele Atari — od konsoli 2600 do Falcona — i muszę przyznać, że mój amigowy światopogląd został poddany ciężkiej próbie.

Jeszcze większą próbą są rozmowy z Tomkiem TDC Cieślewiczem, szczególnie jeśli mamy prowadzić wspólną prezentację, co miało miejsce na imprezie Pixel Heaven w 2017 roku. Słynne "tysiąc i jeden slajdów na każdą okazję" robi swoje, ale w bardzo pozytywnym sensie. Prawie 3 godziny wykładu wystarczyło na zaprezentowanie tylko kilku sprzętów 8-bitowych, ale zainteresowanie i reakcje publiczności były naprawdę zacne. Lubię mówić na temat możliwości technicznych komputerów, a jeśli słuchacze zadają pytania — to już mamy prawie Boże Narodzenie.

À propos, te słowa będzie czytać w grudniu, dlatego chcę Wam złożyć najserdeczniejsze życzenia zdrowia i wszelkiej pomyślności także na Nowy Rok 2020. Bardzo ładna data i myślę, że będzie to też świetny okres dla rozwoju naszych zainteresowań, bo rynek retro cały czas rośnie. Wraz z tym wzrostem następuje większa specjalizacja i prowadzone projekty stają się bardziej dojrzałe. Oby tak dalej!

RETROKOMP
Mikrokomputery wczoraj i dziś

NUMER 15/2019 Wydanie Specjalne

ISSN 2450-5862
Redaktor naczelny: Adam Zalepa

Autorzy tekstów:
Mateusz "Tfardy" Eckert
Krzysztof Kliś
Aleksander Kondała
Piotr „Piter” Krużycki
Marcin Libicki
Piotr "Kroll" Mietniowski
Michał "stRing" Radecki-Mikulicz
Leszek Rosiński
Piotr "Sachy" Sachanowicz
Marcin Skawiński
Kamil Stokowski
Mariusz Wasilewski
Bartosz Wojda

Projekt okładki:
Marzena Bukowska

Korekta: Renata Gralak
Sklad: Andrzej Wilczyński, Adam Zalepa

Wydawca:
A2 Renata Gralak, Łódź

Amigowiec z Atari

ADAM ZALEPA

Moja przygoda z Atari rozpoczęła się dokładnie wtedy, gdy na moim biurku stanął mój pierwszy własny komputer. Wbrew pozorom nie był to produkt z logo wzorowanym na górze Fudżi, lecz... Commodore Plus/4. Od tamtego czasu jestem wierny marce Commodore, ale równolegle poznawałem świat Atari ze względu na zainteresowania mojego kuzyna. W tym samym czasie dysponował on Atari 130XE, a później kolejnymi modelami, aż do Falcona. Czy widziałem wyższość jednej z tych platform? Nie mogłem nie zauważyć różnic, ale niektóre z nich przychodziło mi zaakceptować z nieukrywanym trudem.

Plus/4 nie był pierwszym komputerem, którym miałem okazję używać. Wcześniej pracowałem na wypożyczonym Timexie 2048, który zrobił na mnie piorunujące wrażenie. Pamiętajmy, że były to lata '80-te, w których komputery oglądało się z daleka przez szklaną szybę. Posiadanie jednego z nich w domu było wręcz niesamowitym doświadczeniem. Na dowód przytoczę fakt, że nawet po 30 latach pamiętam zapach wydobywający się z pudełka — styropianów, plastiku i wszystkich innych elementów oryginalnego zestawu.

Dzisiaj brzmi to cokolwiek dziwnie, ale jestem pewien, że ten sentyment pozostanie w mojej głowie do końca życia. Takie to były czasy. Mimo wszystko, romantyczne początki posiadania komputera szybko zamieniły się w codzienną praktykę, gdzie dostrzegałem nie tylko zalety. Dlatego chciałem mieć lepszą maszynę, jeśli tylko nadarzyła się okazja. Commodore Plus/4 to sprzęt bardzo specyficzny, posiadający również bardzo nietypowe oprogramowanie.

Jestem wdzięczny losowi, że dał mi okazję zapoznać się z tym komputerem w okresie rozwoju platform 8-bitowych, bo pewnie później nie potrafiłbym docenić możliwości "plus czwórki". Narzekałem na małą ilość oprogramowania czy niestandardowe porty joysticków, mając jednocześnie przeświadczenie, że sytuacja za granicą jest na pewno lepsza.

W pewnym momencie dowiedziałem się, że kuzyn ma w domu Atari. Wiedziałem o nim sporo z czasopism "Bajtek" i "Komputer", ale była to wiedza wyłącznie teoretyczna. Jedno, czego byłem pewien to fakt, iż z pewnością baza oprogramowania — a szczególnie gier — jest zdecydowanie większa.

Niedługo później dane mi było przekonać się o tym w praktyce i było to bardzo interesujące doświadczenie. Pierwsze, co mnie zdziwiło to wygląd obudowy Atari. Widziałem ją na zdjęciach, ale nigdy nie były zbyt wyraźne. Gazety nie miały takiej jakości jak dzisiaj, poza tym nie kojarzyłem konkretnych modeli komputerów, które posiadały przecież znaczne różnice.

Od razu zauważyłem, że wizualnie 130XE jest o niebo nowocześniejszy od mojego Commodore, szczególnie klawiatura. Wyglądała i działała bardziej profesjonalnie. Kolejna rzecz to stacja dyskietek, której nie

mogłem jeszcze mieć, a nawet widzieć w akcji. Na Atari zobaczyłem wyświetlacz, który pokazywał aktualnie odczytywaną ścieżkę.

Na dodatek dowiedziałem się, że dyskietki mogą się automatycznie uruchamiać i mogę wybierać program do wczytania, bez wpisywania długich komend. Jak na moje ówczesne doświadczenia było to bardzo dużo, a to dopiero początek "wrażeń". Doszło mianowicie do uruchomienia gier.

Na Plus/4 byłem przyzwyczajony do skromnej oprawy, bardzo często nie były nawet podmieniane oryginalne czcionki, a muzyka była raczej symboliczna. Animację obiektów zawsze widziałem jako proste przemieszczanie się z punktu A do punktu B, bez takich ekstrawagancji jak zachowanie bezwładności czy specyficznego toru "ludzika" albo statku kosmicznego. Atari zweryfikowało mój pogląd, bowiem jako jedną z pierwszych gier zobaczyłem "Choplifter".

To było to! Śmigłowiec nie zachowywał się w tak jednoznaczny sposób jak w grach na moim komputerze, a grafika była zdecydowanie bardziej kolorowa niż w większości znanych mi tytułów. Wtedy jeszcze nie zdawałem sobie sprawy, że tę samą grę mogę uruchomić na Plus/4, bo po prostu nie było możliwości jej zdobycia.

Poza tym nawet, gdybym mógł to zrobić, zauważyłbym dużo gorszą grafikę. Jedną z kolejnych gier, którą wczytaliśmy to Ghostbusters. Spodobała mi się latająca "piłka" wskazująca odpowiedni fragment tekstu, tak samo jak samplowany fragment ze śmiechem pochodzącym z filmu. Jednak największe wrażenie zrobiła na mnie muzyka, szczególnie w dalszej części utworu.

Na Plus/4 gry miały w większości proste i krótkie motywy, a więc nie było nad czym się zastanawiać — naciskałem Fire i włączałem samą grę. Na Atari było inaczej i to tutaj pierwszy raz pomyślałem, że przecież to są "normalne" utwory muzyczne! Chciałem nawet słuchać muzyki, zamiast grać, czym wzbudziłem niezadowolenie siedzących obok kolegów. W pewnej chwili zacząłem przyrównywać niektóre dźwięki do brzmienia piosenek zespołu Kombi, co dzisiaj może wydawać się niezbyt trafną uwagą, ale tak to odbierałem.

Gra Choplifter istnieje także w wersji dla Commodore Plus/4, ale dawniej nie miałem o tym wiedzy, nie mówiąc o dostępie do programów.

Kolejną grą, którą pamiętam bardzo dobrze to "Alley Cat". Opowieść o kocie, który baraszkuje na podwórku, brudząc rozwieszone pranie i musi unikać niebezpiecznego psa spuszczonego z łańcucha. Tutaj ponownie spodobała mi się animacja, w pamięć wgryzły mi się też charakterystyczne dźwięki, ale największy problem miałem z... grywalnością.

Mianowicie rozgrywka kosztowała mnie wiele nerwów, bo — o ile doceniłem grafikę — to niestety nie mogłem opanować skakania tytułowego kota, szczególnie na pokrywki od koszy na śmieci. Już nie mówiąc o etapie, gdzie nasz kotek wdziera się do domu, a tam goni go zaczarowana miotła. Na szczęście nie zepsułem joysticka, co nie raz udało mi się na Amidze, ale było to już dużo później. Wracając do Atari, niektóre gry musieliśmy wgrywać z magnetofonu, bo ilość dyskietek była mocno ograniczona.

W tym miejscu zobaczyłem pierwszą ogromną niedogodność, jaką dzisiaj znają wszyscy — szybkość wczytywania programów. Oczywiście nie było tak źle, jak na ZX Spectrum, gdzie wystarczyło tupnąć mocniej nogą, aby pojawił się słynny "LOAD ERROR". Mimo wszystko czekanie 30-40 minut na wgranie gry takiej jak "Bruce Lee" czy "Stealth" przyprawiało o mdłości.

Byłem szczerze zadowolony, że w moim Commodore magnetofon działa zdecydowanie szybciej, chociaż

Monty on the Run — świetna propozycja dla osób lubiących gry labiryntowe, ze wspaniałą muzyką autorstwa Robba Hubbarda.

cały czas miałem na uwadze jakość produkcji dla 130XE. W tym czasie nie zastanawiałem się nad różnymi platformami, nie dumałem o tym, czy warto kupić inny komputer. Naturalną rzeczą była chęć zmiany sprzętu na lepszy model, ale wciąż firmy Commodore.

Z dzisiejszego punktu widzenia jest to bardzo ciekawe, bo przecież kontakt z "lepszym" Atari powinien zaowocować pomysłem sprawienia sobie podobnego komputera. Tak się jednak nie stało, a w zamian zacząłem raczej marzyć o C64. Widziałem go już na jednej z giełd komputerowych w mojej rodzinnej Łodzi, dużo czytałem o jego możliwościach i byłem pewien, iż będzie to model pozwalający nawiązać równorzędną walkę z Atari. I tak się rzeczywiście stało.

Gdy kupiłem C64 i uruchomiłem grę taką jak "Monty on the Run" wiedziałem, że to był dobry krok. Wsłuchiwałem się w temat przewodni, a sama gra nie interesowała mnie tak bardzo jak oprawa audiowizualna, początkowe intro, możliwości uruchomienia opcji nieśmiertelności.

Tych wszystkich "bajerów" brakowało mi na Plus/4, a teraz stały się także moim udziałem. Niestety moja radość nie była długa, bo kuzyn dość szybko zmienił sprzęt na Atari ST. Dokładnie był to model 1040 STFM, którego dziwne oznaczenie traktowałem tak jak opisy silników samochodowych. Zupełnie nie rozumiałem, że oznaczają one określone parametry komputera.

Tak się stało, że osoby dorosłe w rodzinie (czyt. rodzice) przeczytały w Bajtku, że bardzo dobrym sposobem poznawania możliwości sprzętu jest zabawa w programowanie. Jako że mój kuzyn traktował Atari jak konsolę, nie interesowały go dołączone książki opisujące język Basic. Dlatego otrzymałem poważne zadanie: "rozgryźć Atari".
Zabrałem się za uruchomienie Basica i tutaj pierwsze zaskoczenie. TOS wczytuje się bez dyskietki, system jest okienkowy, ale całość dziwnie przypomina połączenie obrazu z Commodore 64 i systemu GEOS. W tym okresie nie mogłem wiedzieć, jak wiele łączy oba komputery, dlatego uznałem to za miły zbieg okoliczności. Jeszcze bardziej zdziwiło mnie sygnalizowanie zawieszenia obrazem "bombek", jak je nazywaliśmy.

Im więcej, tym gorszy błąd wystąpił — tak rozumieliśmy ten komunikat. Atari Basic był dla mnie łatwy, ale nie pisałem w nim rozbudowanych programów. Podobała mi się praca na okienkach, nie rozumiałem natomiast braku możliwości włączenia rozdzielczości typu "High". Z czasem jeden z kolegów pokazał mi "emulator" wysokiej rozdzielczości, ale nie wywołało to we mnie entuzjazmu. Może dlatego, że nie byłem w stanie zrozumieć sensu pracy na ekranie, gdzie w zasadzie widoczna jest co druga linia grafiki.

To był jednak tylko jeden element dużo większej całości moich kontaktów z ST. Każde kolejne odwiedziny rodzinne pozwalały mi odkryć nowe programy dla Atari ST. Jednym z bardziej znaczących był "DeLuxe Paint" - pisany właśnie tak, aby podkreślić elitarność produktu.

Przyznam, że nie pamiętam dokładnej wersji, ale program był na tyle rozbudowany, że odczułem na własnej skórze możliwości graficzne, które rzecz jasna przewyższały mojego poczciwego "komodorka". Szczególnie mocno zwróciłem uwagę na pasek narzędzi w wysokiej rozdzielczości oraz znajdujący się poniżej powiększony fragment grafiki. Rzecz jasna, cały czas oper-

ST SoundTracker — mój pierwszy kontakt z programem muzycznym na 16-bitowe Atari.

owaliśmy na ekranie typu Lowres, ale z punktu widzenia C64, 320 punktów w poziomie z wykorzystaniem wielu kolorów oznaczało najprawdziwszy tryb Hires.

Nie mniejsze wrażenie zrobił na mnie kolejny program graficzny o nazwie "NEOchrome". Mamy tu interfejs podobny do poprzedniego, choć ułożony w nieco inny sposób. Obsługa była dość dziwna, bowiem program doczytuje potrzebne pliki na bieżąco, podobnie jak nowoczesne edytory.

Na twardym dysku możemy się tak "bawić", natomiast wyobraźcie sobie, jak odebrałem takie działanie, widząc to po raz pierwszy na dyskietkach. Stacja dyskietek w ST nie grzeszyła kulturą pracy, więc "NEOchrome" ciągle kojarzy mi się ze zgrzytaniem dyskietki przy uruchamianiu kolejnych funkcji.

Nie było to zbyt miłe doświadczenie, ale doceniłem możliwości programu, w tym — jak wtedy to rozumiałem — funkcję uzyskiwania wielu odcieni tego samego koloru, czego nie mogłem mieć na C64. Zainteresował mnie również program "SoundTracker", który od razu skojarzyłem z podobną nazwą "VoiceTracker".

Ten drugi edytor był pierwszym oryginalnym tytułem, który kupiłem dla mojego Commodore. Wersja dla ST była jednak o wiele lepsza. Nie dość, że ładniejsza z wyglądu, pozwalała na używanie sampli i ogólnie lepszej jakości dźwięku. Jednak na nieszczęście dla Atari, niedługo później w moim zasięgu pojawiła się Amiga 500, która również posiadała własną wersję "SoundTrackera".

W tym okresie zacząłem więc baczniej przyglądać się sprzętowi kuzyna, porównując je z nowym nabytkiem. Dodam, że przez kilka miesięcy posiadałem równolegle C64 i Amigę, co pozwoliło mi na wygłaszanie opinii pod tytułem "tam, gdzie kończą się możliwości Commodore 64, tam Amiga dopiero zaczyna działać".

Szybko dostrzegłem różnice w dźwięku na mojej 500-tce oraz mniejsze obciążenie procesora przy odtwarzaniu modułów. Kilka razy udało się też wczytać bardzo podobne moduły, jak na przykład "Alf Theme" czy "Axel F" i powoli nabierałem coraz większej pewności, że jednak wolę Amigę. No cóż, punkt widzenia zależy od punktu... zaczepienia.

Nie byłbym sobą, gdybym nie sprawdził, jak Atari wypada na tle konkurencji w kategorii elektronicznej rozrywki. Uruchomiliśmy wiele gier, a najbardziej pamiętam wyjazd na wieś i grę w "Lotusa III" na Atari ST dopiero co wypakowanym z Malucha. Ileż to było zabawy! Przeszkadzały nam tylko posiłki serwowane cyklicznie przez rodziców, nie wiadomo po co. Zaraz obok przypominam sobie "Street Fightera 2" oraz "Manchester United".

Wszystkie te gry wyglądały prawie tak samo, jak mojej 500-tce, jednak działały trochę gorzej. Animacja nie zawsze była tak samo płynna, poza tym zdarzało się, że odgłosy uderzeń zastępowały na chwilę fragment muzyki. Te różnice tłumaczyłem sobie 3-kanałowym dźwiękiem ST, w porównaniu do 4 kanałów na Amidze, o czym wyczytałem w jednym z numerów Bajtka.

Nie można było tego nie zauważyć także podczas obsługi "SoundTrackera", ale w okresie C64 trochę inaczej patrzyłem na "duże" Atari, więc pominę to milczeniem. Inne gry, które przykuły mnie na dłużej do ST to produkcje takie jak "Bobo", "North & South", "Terminator 2", "Armour Geddon", "Vroom" czy "Another World".

Co ciekawe, tę ostatnią grę widziałem jako pierwszą na sprzęcie kuzyna, a później na Amidze nie zauważyłem specjalnej różnicy-ani w grafice, ani animacji. Pewnie wynikało to z faktu, że Atari ma procesor taktowany nieco szybciej niż 500-tka, choć nie przypominam sobie, abym wtedy zbyt długo o tym rozmyślał.

Osobną kategorią była dla mnie gra "Dragon's Lair", o której mówiliśmy: film animowany na komputerze. W rozgrywce wcale nie przeszkadzała mała interaktywność bohatera, choć wiele razy joystick był w niezłych opałach. Nie sposób nie wymienić kilku modeli, którymi dysponowaliśmy. U mnie w domu miałem jeden z najprostszych modeli, bez dodatków — Quickshot Python lub Apache. Przy Atari widziałem jednak tak rozbudowane wersje, jak SG Superboard, Aviator czy MegaStar.

Wbrew pozorom można było odczuć ich lepszą jakość, a funkcje typu Timer powodowały, że patrzyłem z zazdrością na lepsze kontrolery. Mimo wszystko gry nie były moim centrum uwagi. Dość szybko chciałem zobaczyć, co Atari potrafi w dziedzinie drukowania. Tak się stało, że w rodzinie pojawiła się drukarka Star LC-200, czyli kolorowa drukarka igłowa.

Okazało się, że można z niej korzystać w programie "Calamus", do którego dołączonych było sporo dyskietek z kolekcją clipartów przygotowanych i rozprowadzanych przez polską firmę. Wtedy już widziałem "Page Streama 2.2" na mojej Amidze, dlatego spodziewałem się podobnego programu. Moim oczom ukazał się jednak bardzo dziwny

Star LC-200 — drukarka, na której drukowałem etykiety do kaset z grami dla C64. W akcji — oczywiście — Calamus na Atari ST.

wygląd, który od razu skojarzyłem z okładkami czasopisma "Komputer". Wcześniej wydawało mi się, że prezentują one raczej oprogramowanie Macintosha, o którym krążyły legendy, a tutaj okazało się, że było to Atari (sic!).

Razem z kuzynem drukowaliśmy głównie etykiety do dyskietek i kaset do C64. Większość moich znajomych miała jeszcze maszyny 8-bitowe, a mówienie o wydruku "dopiero co spod igły" robiło wrażenie. Cieszyliśmy się wysoką rozdzielczością i możliwością wybierania nie tylko ciekawych obrazów z dyskietek, ale także stosowaniem wielu kolorów.

Nikomu nie przeszkadzała mała prędkość działania "Calamusa", dla mnie natomiast obsługa tego programu zawsze pozostawała, i pozostała do dziś, jedną wielką niewiadomą. Być może jestem zbyt ostry w tej opinii, ale używałem wielu programów DTP na różnych komputerach i prawie zawsze pasek narzędzi jest dla mnie czytelny.
W "Calamusie" widoczne ikony nie kojarzą mi się z niczym, a nawet trudno mi zapamiętać poszczególne opcje. Zdaję sobie sprawę, że wychodzi ze mnie teraz brak tolerancji dla nietypowych rozwiązań, przyjmuję również, że program ma duże możliwości, ale nie wiem czy chciałbym go używać na stałe.

Ponadto jakoś nie byłem w stanie sobie wyobrazić profesjonalnej pracy na monochromatycznym ekranie, bo na Amidze nigdy nie było problemów z kolorami, choć tryb Interlace nie był zbyt przyjemny. Mógłbym opowiadać jeszcze bardzo długo o moich doświadczeniach z "małym" i "dużym" Atari, ale myślę, że powyższe rozważania pokazują mój punkt widzenia.

Mówiłem kiedyś, że jestem "nieuleczalnym" Amigowcem i to chyba nigdy się nie zmieni. Doceniam różne platformy, nie neguję ich

możliwości, specyfiki i uroku, ale jednak system operacyjny Amigi, a także jej super płynne gry i świetny dźwięk są dla mnie nie do zastąpienia. Cieszę się, że mogłem korzystać z wielu różnych maszyn w okresie ich największej popularności i obserwować rozwój.

Szkoda tylko, że kilka lat później te wszystkie osiągnięcia zostały zdewaluowane przez rynek peceta. Nigdy nie zapomnę czasów, gdy komputery określały ich faktyczne możliwości, a nie marketing. Kiedy konkurowaliśmy pomiędzy sobą na funkcje programów, systemów Turbo czy szybkość działania.

W erze sprzed dominacji PC nawet pecetowcy uczestniczyli w tych porównaniach, nie widząc nic złego w tym, że sprzęt z definicji "domowy" może prześcignąć "osobisty" w niektórych miejscach. Uważam, że Atari miało wielki udział nie tylko w popularyzacji informatyki w Polsce — był to przecież sprzęt jak żaden inny dostępny w sklepach i promowany oficjalnie przez czasopisma. Z perspektywy czasu największą wartość widzę jednak w pokazaniu, że komputer używany w domu może być faktyczną alternatywą dla rodzącego się imperium spod znaku IBM PC.

Amidze nigdy nie udało się osiągnąć takich celów, choć udało się jej coś innego — skierować wzrost sprzedaży komputerów w swoją stronę. Według dzisiejszych informacji, przez krótki czas krzywa popytu i podaży zmniejszała się tym bardziej przy PC, im bardziej wzrastała przy Amidze. Takiego stanu rzeczy nie utrzymano w dłuższym okresie czasu, ale tutaj mam refleksję.

QuickShot Aviator — jeden z najciekawszych joysticków, polecany do symulatorów lotu.

Rynek sprzętu "domowego" powinien próbować skonsolidować się podobnie jak "profesjonalny". Być może takie działanie mogłoby zatrzymać na rozsądnym poziomie udział rynkowy peceta. Tak się niestety nie stało, a teraz możemy tylko spekulować, wspominać i żałować.

Na koniec chciałbym przypomnieć, komu zawdzięczamy zarówno rozwój firmy Commodore, jak i Atari. Ten sam człowiek stał za popularnością C64 i Atari. Mówię oczywiście o Jacku Tramielu, a pisząc bardziej po polsku — Idku Trzmielu, który zrobił rzecz niesłychaną. Jako żydowski emigrant urodzony w Łodzi, z tragiczną kartą historii, między innymi pobytem w obozie w Oświęcimiu, nie poddał się. Wyjechał do USA i stworzył jedną z największych firm komputerowych w historii.

Gdyby nie on, nie byłoby C64, nie usłyszelibyśmy o komputerze takim jak "Commodore Amiga" oraz nie moglibyśmy doświadczyć specyfiki Atari ST. Uważam, że właśnie tacy ludzie powinni być wymieniani w annałach historii i to o nich powinni dowiadywać się przede wszystkim uczniowie szkół i słuchacze uczelni wyższych.

Jack Tramiel zmarł tylko 5 lat temu, w 2012 roku i w moim przekonaniu przez całe swoje życie nie został odpowiednio doceniony za to, co zrobił dla komputerowego świata. My jednak możemy zrobić coś więcej. Najlepszym sposobem uczczenia jego pamięci jest używanie produktów, które stworzył. I tego się trzymajmy.

Pierwszy raz: gry na ST

ADAM ZALEPA

Muszę się Wam do czegoś przyznać — jestem niereformowalnym Amigowcem. Większość mojego komputerowego życia poświęciłem tej platformie. Jednak to nie znaczy, że nie doceniam innych komputerów. Wręcz przeciwnie, Atari jest w mojej głowie ważnym sprzętem, choć wracam do niego po naprawdę wielu latach różnych perturbacji. Za to bez emulacji — tylko "pure Atari".

Nie mogę przecież nie zwracać uwagi na sprzęt, w którym miał swój udział także Jay Miner, a później twórca Commodore 64, czyli Jack Tramiel. Obaj panowie są dla mnie wzorem wizjonerstwa w informatyce, niezależnie od okoliczności, które nie pozwoliły im rozwinąć skrzydeł do końca. W tym artykule chcę pokazać, co zwróciło moją uwagę na Atari w czasach, gry chodziłem do szkoły podstawowej, a wszelkie różnice pomiędzy komputerami były ważniejsze od ocen z matematyki czy języka polskiego.

O czym chcę napisać? Oczywiście chodzi o gry. Nie pamiętam, kiedy zobaczyłem pierwszą grę na Atari. Pamiętam za to, jakie wrażenie zrobiła na mnie klawiatura modelu 130 XE w porównaniu do mojego Commodore 64. Rozbudziło to moje zainteresowanie komputerami spod znaku góry Fuji-san i tylko przypadek sprawił, że na moim biurku pojawiały się kolejne 8-bitowe modele Commodore. Amiga była już świadomym wyborem, ale o tym możemy dyskutować godzinami.

Wracając do gier, moja rodzina zdobywała kolejne tytuły z różnych źródeł i niekoniecznie były to bardzo znane produkcje — przynajmniej z dzisiejszego punktu widzenia. Pisałem kiedyś o grach, które zapamiętałem najbardziej — teraz czas na przedstawienie tych, które mnie najbardziej zadziwiły, a wśród użytkowników C64 wcale nie były tak bardzo popularne.

Pierwszym tytułem jest "Eidolon". Strzelanina trójwymiarowa dla 8-bitów! Prawdziwe cudo swoich czasów, bo przecież gra została wydana już w 1985 roku. Znałem ją z Commodore, ale nigdy nie wzbudziła we mnie większego zainteresowania, bo działała bardzo wolno. Dość szybko odkryłem, że na Atari wygląda to zupełnie inaczej. Paleta kolorów jest jakby lepiej dobrana już od pierwszego logo "Lucasfilm Games". Później jest tylko lepiej, bowiem cały panel w dolnej części ekranu nabiera życia.

I oczywiście — szybkość. W porównaniu do C64 wtedy była to dla mnie kolosalna różnica. Dzisiaj oczywiście też ją widać i szczerze mówiąc — nabieram coraz większej ochoty do zakupu kilku modeli Atari. Przynajmniej 8-bitowych, bo w innych sprawach jeszcze nie jestem przekonany. Nie będę wchodził w techniczne szczegóły, ale na Atari gra "Eidolon" jest prawie płynna, natomiast C64 trochę tu kuleje. Gdy jesteśmy przy grach 3D, warto wspomnieć o Najemniku, czyli "Mercenary".

Tu mamy trochę odwrotny problem — mianowicie na Commodore animacja jest bardziej płynna, choć nie

ma takiej różnicy jak przy "Eidolon". Za to paleta barw jest znowu bardziej atrakcyjna na Atari. Osobiście nie mam nic przeciwko symulowaniu przestrzeni za pomocą kolorów zielonego i niebieskiego, ale wersja dla Atari wygląda bardziej realistycznie ze względu na przytłumione kolory.

Trudno to może wyjaśnić, lecz jest to kolejny tytuł, który zainteresował mnie na 130 XE, a na C64 uruchamiałem go bez większego zaangażowania. Kolejna sprawa to panel na dole ekranu, który w obu wersjach na podobne kolory. Jak widać "prawie" robi naprawdę dużą różnicę i niestety — Atari robi lepszą robotę.

Nie byłbym sobą, gdybym w tym momencie nie porównał tej samej gry w wersjach dla Amigi i Atari ST. Nie widać dużych różnic, ale tu odzywają się już moje preferencje 16-bitowe. Panel na Przyjaciółce jest nieco większy, a animacja bardziej płynna. Mimo wszystko, spoglądając na grę bardziej obiektywnie — nie ma tu takiej różnicy jak w przypadku wersji dla 8-bitowców. "Mercenary" mamy także w wersji dla innych komputerów, na przykład ZX Spectrum, więc możemy mieć pełne spektrum porównania.

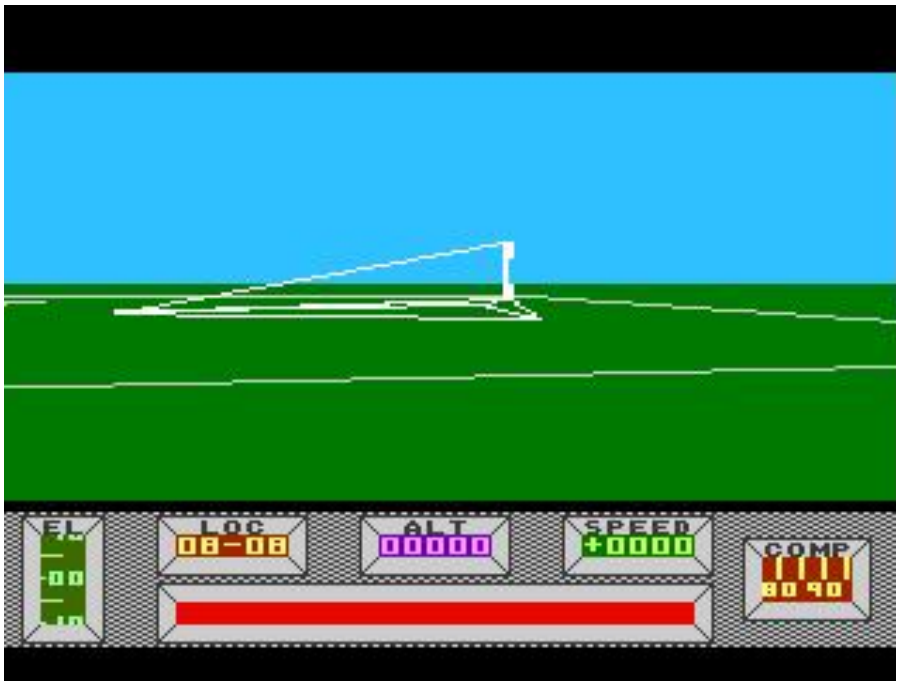
Polecam szczególnie odnaleźć logo Commodore, które składa się jak domek z kart. Co ciekawe, w wersji dla Spectrum gra jest dość płynna, a kolory wcale nie są dobrane dużo gorzej niż na C64. Ekran rozgrywki jest nieco mniejszy, ale po paru minutach nie ma to większego znaczenia. W każdym razie — gdybym miał "Mercenary" w wersji dla ZX Spectrum w latach '80-tych, z pewnością byłbym wniebowzięty. Przejdźmy teraz do nowszych czasów, mianowicie 2007 roku, gdy wydana została gra "Crownland". Odkryłem ją niedawno i... szczęka mi opadła. Kolorystyka, animacja, poziom skomplikowania lokacji, dynamika rozgrywki, efekt paralaksy i inne elementy robią kolosalne wrażenie.

W moim przekonaniu gra jest 8-bitowym odpowiednikiem platformówek, które pojawiały się na Amidze. Spotkałem się nawet z komentarzami, że ludzie traktują 8-bitowe Atari właśnie jak 8-bitową odmianę Przyjaciółki.

Oczywiście mamy mniejszą rozdzielczość, zupełnie inny dźwięk i mniejszy ekran gry, ale "Crownland" pod względem grywalności może śmiało konkurować ze swoimi "starszymi" odpowiednikami. Kto nie zna tej gry, polecam! Kolejna gra - "Project-M", czyli "Wolfenstein 3D" na 8 -bitowe Atari. Hm, tu mam wrażenia zupełnie innego rodzaju. Gra jest wybitnie płynna, ale jednak trzeba zwrócić uwagę, że obraz generowany jest co drugą linię. Jednak samo to, że autor doprowadził do tak dobrego poziomu wydajności silnika, zasługuje na docenienie.

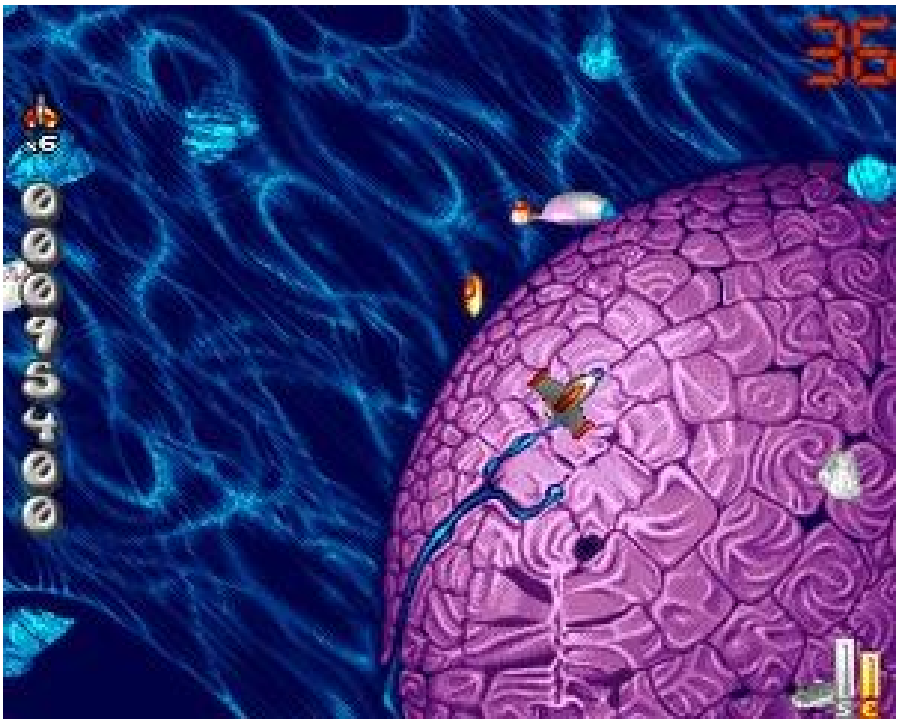
Amigowa wersja "Wolf3D" w miarę płynnie działa dopiero na procesorze 68030/25 MHz. Oczywiście znowu mamy do czynienia z wyższą rozdzielczością, większą ilością kolorów, ale biorąc pod uwagę Atari — gra powinna działać na zwykłej Amidze 500. Czy da się to kiedyś osiągnąć? Nie miałbym wielkich zastrzeżeń. gdyby nie fakt, iż widziałem także wersję "Wolfensteina" działającą na Atari STE z 2 MB pamięci.

Nie ma co ukrywać, jest bardziej płynny, a do tego wizualnie zdecydowanie bardziej dopracowany. Środowisko fanów Atari ciągle zaskakuje mnie współpracą i możli-



Mercenary w wersji dla Atari dawał zupełnie inną kolorystykę otoczenia niż na moim Commodore 64.

Porównanie: gra Star Dust w wersji na Atari STE (na górze) oraz Amigę 500 (po prawej).



wościami optymalizacji algorytmów, ale nie sądziłem, że różnica będzie aż tak bardzo widoczna.

Oczywiście trudno porównywać poszczególne silniki, bo są to przecież zupełnie różne autorskie rozwiązania działające na innych architekturach sprzętowych. Jednak gracza interesuje efekt końcowy, który wypada na Atari lepiej. Na koniec chciałbym wypowiedzieć się w sprawie bardzo popularnej gry "Star Dust", która w wersji dla Amigi 500 w latach '90-tych zrobiła prawdziwą furorę.

Od pewnego czasu mamy konwersję na Atari STE, która wygląda bardzo dobrze, ale nie jest pozbawiona kilku wad. Po pierwsze intro nie zawiera przelatującego napisu z tytułem gry. Trochę to zaburza wprowadzenie, bowiem jest ono wzorowane na Gwiezdnych Wojnach, a bez "głównego logo" trochę tracimy. Za to animacja żółtych napisów jest nieco bardziej płynna na Atari, choć trzeba się bardzo przyjrzeć, aby zauważyć różnicę. Na końcu intro amigowa wersja zawiera krótką animację świetnie wykonanego myśliwca kosmicznego, który powodował niezapomniane wrażenia, szczególnie podczas oglądania "Star Dust" po raz pierwszy. W wersji na Atari ekran po prostu jest wygaszony i tego elementu nie ma wcale. Wielka szkoda, ale i satysfakcja po mojej stronie — czyżby Atari w tym wypadku "nie dało rady"?

Kolejny element — wybór poziomów, czyli "War Plan". Nie dość, że na STE jest mniejszy to dodatkowo pozbawiony kolorów, a więc sprawdza się pogląd, że amigowa feeria barw nie jest łatwa do podrobienia. To samo dotyczy grafiki w poszczególnych poziomach, która ma inną paletę barw, ale nie jest to specjalnie drażniące, chyba że chcemy koniecznie porównywać atarowski "Star Dust" do oryginału. Podoba mi się natomiast przeniesienie punktacji i oznaczenia ilości "żyć" z boku na górę. Panel jest wykonany estety-

cznie i doskonale wzbogaca dość pustą część ekranu.

Co prawda gra traci przez to swój unikalny wygląd, ale za to oznaczenia typu "Energy" czy "Shield" są bardziej czytelne niż litery "S" i "C" w wersji amigowej. Animacja de facto jest bardzo podobna, rozgrywka jest tak samo dynamiczna i płynna, ale całość odbywa się na trochę mniejszym ekranie.

Zupełnie inne wrażenie robią słynne tunele, czyli poziomy przejściowe pomiędzy "światami" w grze. Na Amidze mamy zachowany pseudotrójwymiarowy wygląd, obszar rozgrywki nie jest zmniejszany, a ruchy naszym statkiem kosmicznym powodują zmianę perspektywy tunelu w bardzo atrakcyjny sposób. Swojego czasu byłem zafascynowany tym elementem i zastanawiałem się "jak oni to zrobili na zwykłej 500-tce?".

Niestety tunel na Atari wygląda całkiem inaczej. Po pierwsze ekran jest już dużo mniejszy — o jakieś 20% wysokości. Druga sprawa to wygląd, który przypomina bardziej znaki z Matrixa niż okrągły tunel na Przyjaciółce. Zmiana pozycji gracza nie powoduje tak dynamicznych zmian perspektywy, mniej zróżnicowana jest także kolorystyka poszczególnych elementów.

Podobnie jak wcześniej, mamy natomiast panel, którego brakuje na Amidze. Powoduje on większą spójność rozgrywki, bowiem gracz widzi podobne oznaczenia w poziomach "zwykłych" oraz "przejściowych". Co prawda odliczający się czas jest dużo mniej widoczny, więc emocje z przechodzenia do kolejnego

Super Star Dust na Amidze 1200 działa częściowo w rozdzielczości Hi-res, czyli 640x256 punktów.

"świata" są mniejsze, niemniej wygląda to bardzo dobrze.

Po ukończeniu tunelu na ekranie pojawia się podsumowanie, punktacja oraz fajerwerki w tle. Atari pokazuje tu mniejszą płynność, a samej rozbłyski są mniejsze i mniej kolorowe.

Prawdę mówiąc, bardziej spodziewałem się różnic w tunelach, ale nie bądźmy też purystami — różnica nie jest wielka. "Star Dust" jest przykładem, jak wiele można wycisnąć z STE, gdy projektu podejmą się odpowiednie osoby.

Na Amidze możemy zagrać też w "Super Star Dust", który działa w rozdzielczości 640x200, ale to już zupełnie inny jakościowo produkt. W każdym razie atarowską konwersję oceniam bardzo pozytywnie, choć widzę niedostatki w porównaniu z pierwowzorem działającym przecież na zwykłej Amidze 500 z 1 MB pamięci.

Powyższe gry są dla mnie ważne, ale i pokazują dwa oblicza obu platform. Z jednej strony optymalizacja algorytmów dla "małego" Atari powoduje, że trudno znaleźć odpowiedniki dla Commodore 64. Z drugiej — Amiga pokazuje pazurki tam, gdzie mamy mieć szaloną akcję i jak najwięcej kolorów na ekranie. Pokazuje to, że obie architektury mogą się wzajemnie uzupełniać, przynajmniej jeśli chodzi o oprogramowanie rozrywkowe. Dla mnie — Atari, Commodore i Amiga przenikają się wzajemnie przez pryzmat pomysłów Jaya Minera i Jacka Tramiela. Nie wszystkie zostały zrealizowane we właściwym czasie, ale dziś możemy je ocenić bardziej obiektywne i z właściwym dystansem.

Lepsza kontrola grafiki

MARCIN LIBICKI

Atari posiada 9 podstawowych trybów graficznych. Tryby od 1 do 8 mają dzielony ekran, jednak może być to zmienione przez dodanie 16 do numeru trybu. Tryby 1 i 2 są trybami tekstowymi z pięcioma kolorami. Znaki w trybie graficznym 1 są dwukrotnie wyższe i dwukrotnie szersze niż w trybie 0. Znaki w trybie 2 są natomiast dwa razy wyższe i dwa razy szersze niż w trybie 0.

Zestaw znaków w trybie graficznym 0 ma 128 znaków, duże i małe litery, znaki interpunkcyjne, cyfry i znaki graficzne Atari. Jednak w trybach graficznych 1 i 2 dostępne są tylko 64 znaki jednocześnie. Dostępne są tu trzy możliwości: cyfry, litery i znaki interpunkcyjne, znaki graficzne Atari lub można zaprojektować własny zestaw znaków.

Na pewno często używasz zarówno znaków Spacji, jak i znaków graficznych. Istnieją dwa sposoby tworzenia pustych Spacji. Jednym z nich jest rezygnacja z jednego z pięciu dostępnych kolorów. Wystarczy, że rejestr koloru "zerowego" będzie miał taki sam kolor jak tło, a następnie będziesz wyświetlał inne znaku przy użyciu rejestrów kolorów 1, 2 i 3. Jest to bardzo proste rozwiązanie.

Druga metoda wymaga więcej pracy, bowiem jeden znak musi być redefiniowany. Jak to zrobić? Po pierwsze, musimy przydzielić miejsce w pamięci RAM dla zestawu znaków i zabezpieczyć go przed dostępem Basica.

Górna część pamięci RAM to koniec części pamięci dostępnej dla użytkownika i jest przechowywana w miejscu zwanym RAMTOP. Obszar powyżej wartości zapisanej w pamięci RAMTOP to ROM, który zawiera stałe, które nie mogą zostać zmienione. Tutaj zapisywany jest także system operacyjny.

Jeśli w pamięci RAMTOP zapisujemy niższą wartość, system operacyjny uzna, że dostępna jest mniejsza ilość pamięci RAM. W ten sposób możemy uchronić nasz nowy zestaw znaków przed zmianą lub skasowaniem. Dlatego najpierw trzeba zarezerwować pamięć dla nowego zestawu znaków.

Tryby graficzne 1 i 2 wymagają 512 bajtów lub dwóch stron do redefiniowania zestawu znaków. W trybie 0 potrzebujemy 1024 bajtów lub czterech stron, aby redefiniować 128 dostępnych znaków.

Następnie zmieniamy lokalizację obecnego zestawu znaków z pamięci ROM do zarezerwowanej części pamięci. Można to łatwo osiągnąć za pomocą pętli FOR/NEXT oraz instrukcji PEEK.

Zestaw znaków zawierający duże litery, cyfry i znaki interpunkcyjne znajduje się pod adresem 57344 w pamięci ROM, a alternatywny zestaw znaków graficznych pod adresem 57856.

Dalej należy poinformować system operacyjny, w którym miejscu znajduje się nowy zestaw znaków. Robimy to za pomocą instrukcji POKE 756,X, gdzie X oznacza adres nowego zestawu znaków.

Za każdym razem, gdy wykonywana jest instrukcja graficzna lub reset, wartość pod adresem 756 jest rese-

towana do wartości 224, czyli adresu strony startowej starego zestawu znaków (zapisanego w pamięci ROM). Z tego względu najlepiej dopisać wspomnianą instrukcję POKE po każdej instrukcji trybu graficznego.

Teraz już możemy zdefiniować własne znaki. Definicja znaku używa 8 bajtów w pamięci, natomiast w celu zastąpienia dotychczasowego kształtu musimy użyć 8 cyfr. Jeśli pod danym adresem zapiszemy wartość zero, będziemy mieli puste miejsce.

Programy przedstawione w ramkach wykonują omówione powyżej operacje, a w zrozumieniu działania powinny pomóc Wam dołączone komentarze. Mam nadzieję, że zainspiruje to Was do własnych eksperymentów z pamięcią Atari.

```
Program nr 1

110 GRAPHICS 1:POKE 756,226
120 ? "THIS IS WHAT HAPPENS WHEN 756 IS POKED WITH 226 IN GR. 1"
130 FOR WAIT=1 TO 2000:NEXT WAIT
140 SETCOLOR 0,0,0:REM SET COLOR REGISTER 0 TO SAME COLOR AS BACKGROUND
150 ? "THIS IS WHAT HAPPENS WHEN A COLOR REGISTER IS MADE SAME COLOR AS BACKGROUND."
160 FOR WAIT=1 TO 2000:NEXT WAIT

Program nr 2

120 POKE 106,PEEK(106)-2
130 GRAPHICS 2+16:REM GR.STMT.HERE PREVENTS OVERLAP OF DISPLAY LIST & CHARACTER SET
140 REM STEP TWO MOVE: CHARACTER SET INTO NEW LOCATION
150 A=PEEK(106)*256
160 FOR B=0 TO 511
170 POKE A+B,PEEK(57856+B)
180 NEXT B
190 REM STEP THREE: POKE NEW ADDRESS OF CHARACTER SET
200 POKE 756,PEEK(106)
210 REM STEP FOUR: CHANGE HEART TO BLANK SPACE
220 FOR C=0 TO 7
230 POKE A+C,0
240 NEXT C

310 REM SET UP COLOR REGISTERS
330 SETCOLOR 0,13,8:REM GREEN
340 SETCOLOR 1,4,8:REM PINK
350 SETCOLOR 2,10,8:REM TURQUOISE
360 SETCOLOR 3,2,8:REM GOLD
365 SETCOLOR 4,12,4:REM BACKGROUND COLOR TO GREEN

390 COLOR 60:PLOT 5,5:REM GREEN ARROW
400 COLOR 28:PLOT 6,5:REM PINK ARROW
410 COLOR 188:PLOT 7,5:REM TURQUOISE ARROW
420 COLOR 156:PLOT 8,5:REM GOLD ARROW
450 GOTO 450:REM KEEPS DISPLAY ON SCREEN
```

Nie tylko Atari DOS

KAMIL STOKOWSKI

Przy okazji numeru specjalnego chciałbym poruszyć temat atarowskiego DOS-u. Wiele osób nie zdaje sobie sprawy, że komputery 8-bitowe to nie tylko Basic, a małe Atari wyróżnia się właśnie obsługą stacji dyskietek. Atari DOS w zasadzie nie jest systemem operacyjnym, lecz programem odpowiadającym za zarządzanie plikami na dyskach, ale w obsłudze przypomina prawdziwy "dyskowy OS".

ATARI DOS

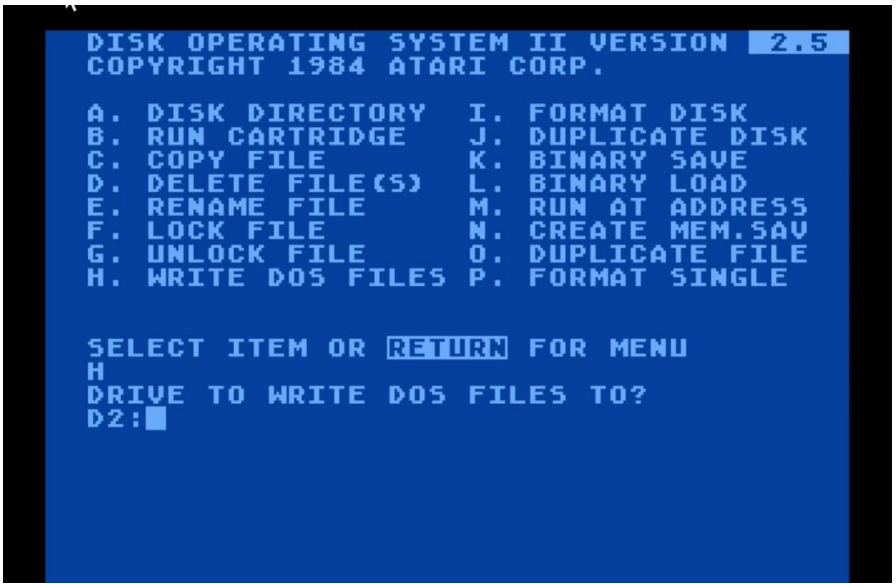
Zacznijmy od tego, co wydaje się oczywiste. Atari DOS — jak każdy tego typu system — przechowuje dane w plikach, a każdy z nich zawiera jedną konkretną grupę informacji. Pliki mogą być zapisywane losowo na dysku lub mogą być pogrupowane w katalogi.

Każdy dysk posiada co najmniej jeden katalog zwany katalogiem głównym, który zawiera wszystko, co znajduje się na dysku. Można tworzyć inne katalogi i umieszczać w nich pliki. Teraz już dochodzimy do informacji bardziej szczegółowych. Otóż katalog może pomieścić maksymalnie 1250 plików.

Oczywiście podkatalogi mogą zawierać jeszcze więcej plików i kolejnych katalogów. Podczas tworzenia pliku lub katalogu należy nadać mu nazwę, która może mieć do 8 znaków, a następnie, jeśli chcemy, może być to dodatkowo kropka i rozszerzenie (długość do 3 znaków). Ważne, że rozszerzenie nie jest konieczne, aby zapisać informacje.

Znaki w nazwie mogą być literami, cyframi lub znakami specjalnymi, jak na przykład podkreślenie czy "@". Wszystkie litery wpisywane małymi literami są zamieniane na wielkie litery.

W nazwie lub rozszerzeniu nie dopuszcza się jednak stosowania żadnych znaków interpunkcyjnych ani



Atari DOS nie jest systemem operacyjnym w pełnym znaczeniu tego słowa.

K-DOS, czyli alternatywa dla oryginalnego Atari DOS-u.

symboli. Rozszerzenie jest przydatne w przypadku plików związanych z daną nazwą, ale różniących się od siebie, bowiem można nimi zarządzać jako grupa. Gdy odnosisz się do pliku, musisz wskazać, na którym dysku i w jakim katalogu został zapisany, a także podać jego nazwę. Jest to normalna sprawa w jakimkolwiek DOS-ie, ale trzeba pamiętać o zasadach tworzenia ścieżki dostępu.

Nazwy urządzeń używane przez Atari DOS to "D" dla normalnych dysków i "A" dla dysków w formacie DOS 2.0 lub 2.5. Inne urządzenia mogą być również używane, takie jak "P:" (drukarka) lub "E:" (edytor ekranowy). Numer urządzenia to po prostu numer napędu dyskowego. Nazwy ścieżek nie mogą przekraczać 80 znaków. Ogranicza to ilość poziomów podkatalogów, ale jest na to prosta rada.

Jeśli potrzebujesz więcej poziomów — użyj krótszych nazw katalogów. DOS zapewnia możliwość stworzenia skrótu podczas pracy z długimi ścieżkami. Pozwala to na zdefiniowanie czegoś w rodzaju katalogu roboczego, którego nazwa może być zapamiętana.

Podczas pierwszego uruchomienia jest to katalog główny, ale można zdefiniować własny za pomocą opcji "Working Directory" w menu. Rozpoznawane są też filtry, czyli znaki specjalne zastępujące dowolne znaki w nazwach plików. Umożliwiają odwołanie się do grupy plików, a nie do każdego z nich z osobna. Na Atari może być to znak zapytania lub gwiazdka — pierwszy oznacza pojedynczy znak, a drugi — dowolną ilość znaków w nazwie lub rozszerzeniu.

K-DOS

K-DOS firmy K-Byte jest alternatywą dla oryginalnego systemu operacyjnego Atari. Jest kompatybilny z systemem Atari DOS, ale oferuje większy poziom kontroli nad urządzeniami peryferyjnymi i pamięcią. Oferuje wiele funkcji, które mogą być docenione przez mało zaawansowanego użytkownika Atari, a jednocześnie ma też wiele udogodnień dla programisty.

Niestety K-DOS zajmuje większą ilość wolnej pamięci. Jest rezydentny, więc większość jego funkcji jest natychmiast dostępna. Zamiast ok. 40 kilobajtów pamięci użytkownik otrzyma do dyspozycji tylko niecałe 25 kilobajty. W zamian K-DOS oferuje wiele atrakcyjnych funkcji — przykładowo, jego funkcje są dostępne z poziomu języka Basic. K-DOS jest sterowany za pomocą poleceń, więc nie trzeba wywoływać żadnych menu.

Wystarczy poprzedzić polecenie przecinkiem lub innym znakiem, który można zdefiniować, a funkcja DOS jest wykonywana bez zmiany środowiska. Składnia wymagana dla linii poleceń jest elastyczna: przecinki mogą być zastąpione spacjami, dopuszczalne jest używanie małych liter. Komunikaty o błędach pojawiają się w formie tekstowej, w języku angielskim — nie ma tu nieczytelnych numerów błędów, które trzeba zapamiętać.

Innym aspektem sterowania systemem, jakie oferuje K-DOS jest fakt, że pozwala na zatrzymanie operacji wejścia/wyjścia na dysku po prostu przez naciśnięcie klawisza BREAK — i to bez utraty danych. Wprowadza także kilka zmian w oryginalnych funkcjach DOS.

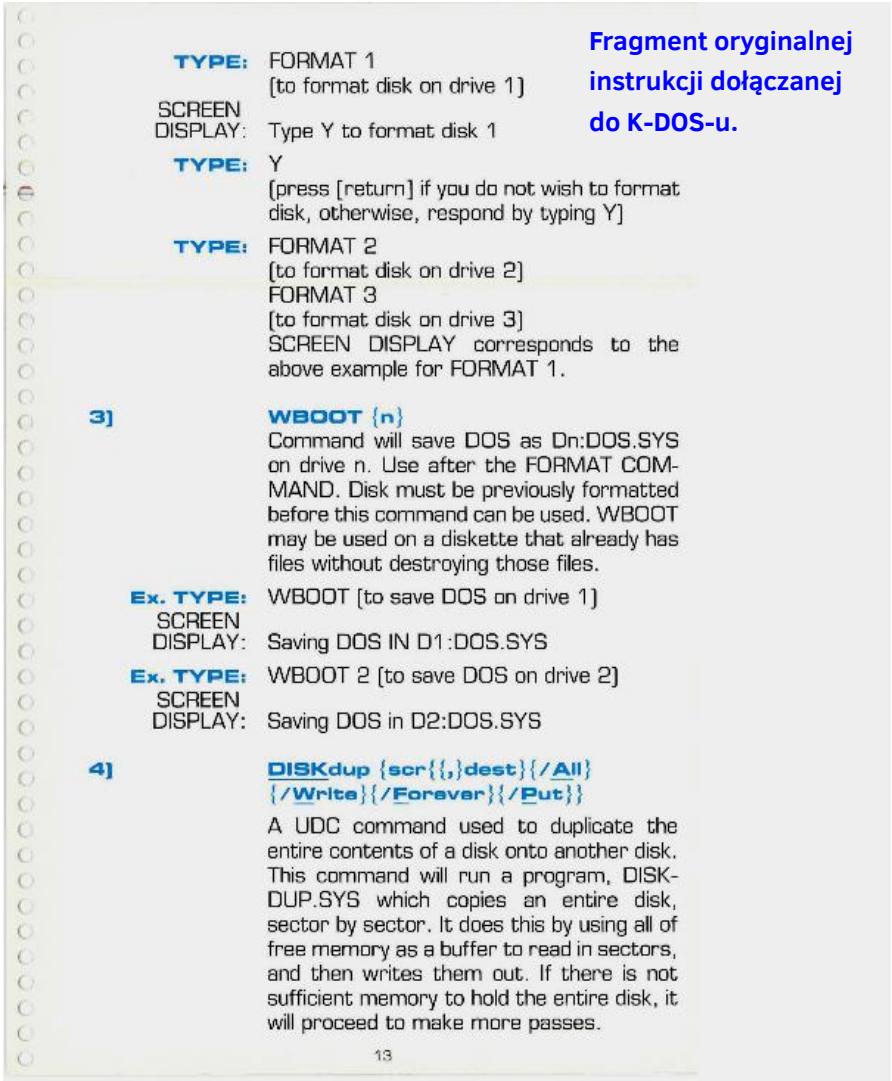
Na przykład, INIT łączy w jednej operacji formatowanie i zapis plików DOS-u na nowy dysk. Jednocześnie obie funkcje są nadal dostępne oddzielnie.

Funkcja DISKDUP pozwala na szybszy zapis bez konieczności weryfikacji i ciągłego ponownego testowania uszkodzonych sektorów. Istnieje też osobne polecenie AP-

PEND, które umożliwia wprowadzanie danych na końcu pliku bezpośrednio z klawiatury. Funkcja APPEND wykorzystuje do- wolne miejsce dostępne w ostatnim sektorze, zamiast używać nowego sektora, jak robi to Atari DOS.

Warto zauważyć, że K-DOS nie zajmuje całej wolnej pamięci. Zawiera również kompletny monitor języka maszynowego, który pozwala na analizę zawartości pamięci, stosując liczby w formacie szesnastkowym i lub znaki ASCII, a także badanie i zmianę zawartości rejestrów. Istnieją dwa sposoby na wykonanie programu w języku maszynowym.

Funkcja GO uruchamia program po zamknięciu wszystkich urządzeń i nie zachowuje rejestrów. Druga możliwość to funkcja PROCEED, która kontynuuje program po osiągnięciu punktu przerwania, ale nie zmienia zawartości rejestrów ani statusu żadnego urządzenia. Dlatego jest to opcja bardzo wygodna przy debugowaniu.



TYPE: FORMAT 1
(to format disk on drive 1)

SCREEN DISPLAY: Type Y to format disk 1

TYPE: Y
(press [return] if you do not wish to format disk, otherwise, respond by typing Y]

TYPE: FORMAT 2
(to format disk on drive 2]
FORMAT 3
(to format disk on drive 3]
SCREEN DISPLAY corresponds to the above example for FORMAT 1.

3] **WBOOT {n}**
Command will save DOS as Dn:DOS.SYS on drive n. Use after the FORMAT COMMAND. Disk must be previously formatted before this command can be used. WBOOT may be used on a diskette that already has files without destroying those files.

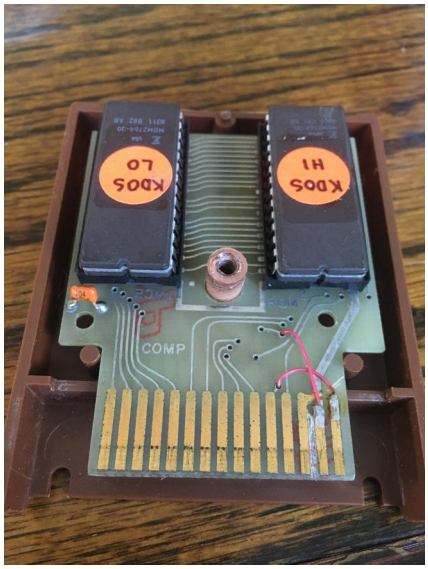
Ex. TYPE: WBOOT [to save DOS on drive 1]
SCREEN DISPLAY: Saving DOS IN D1:DOS.SYS

Ex. TYPE: WBOOT 2 [to save DOS on drive 2]
SCREEN DISPLAY: Saving DOS in D2:DOS.SYS

4] **DISKdup {scr{,}dest}{/All}{/Write}{/Forever}{/Put}}**
A UDC command used to duplicate the entire contents of a disk onto another disk. This command will run a program, DISKDUP.SYS which copies an entire disk, sector by sector. It does this by using all of free memory as a buffer to read in sectors, and then writes them out. If there is not sufficient memory to hold the entire disk, it will proceed to make more passes.

13

Fragment oryginalnej instrukcji dołączanej do K-DOS-u.



Cartridge K-DOS po zdjęciu obudowy.

Ponadto, K-DOS oferuje wiele poleceń, które umożliwiają dostęp do procedur używanych wewnętrznie przez DOS. Na przykład, COLD i WARM zapewniają łatwy sposób na miękki lub twardy reset.

Jako osoba, która używa swojego komputera głównie do programowania, uważam, że K-DOS jest szczególnie pomocny w pisaniu programów, które łączą język Basic z podprogramami języka maszynowego. W pozostałych przypadkach lepiej użyć Atari DOS, dzięki któremu uzyskamy dużo więcej wolnej pamięci.

Dla Atari opracowano wiele DOS-ów. Wiele z nich zostało wydanych przez producentów stacji dyskietek, którzy wprowadzali własne poprawki.

Systemy te mogły często odczytywać nośniki o większej gęstości, a także korzystać z technik typu Warp Speed lub Ultra Speed przyspieszających transmisję danych. Większość DOS–ów jest zgodna z Atari DOS 2.0, wyjątkiem jest Sparta DOS.

Ciągi tekstowe w Basicu

Opracował:
MARCIN LIBICKI

Atari Basic różni się od większości innych dialektów Basica pod względem obsługi ciągów tekstowych. Pozwala na łańcuchy dowolnej długości, ograniczone tylko wolną pamięcią. Jednocześnie wyrażenie takie jak A$(X,Y) w Atari Basic jest odniesieniem do fragmentu ciągu A$ zaczynającego się na pozycji X i kończącego się na pozycji Y. W wielu innych Basicach podobne wyrażenie jest odniesieniem do tablicy ciągów, co stanowi trudność w konwersji programów.

Używanie części łańcuchów jest wygodne, bo pozwala wyeliminować powolne funkcje, takie jak MID$(), LEFT$(), RIGHT$() na rzecz kompaktowych, bezpośrednich odniesień do konkretnych zmiennych, jak A$. Działa to szybciej, a sam program może być krótszy.

Jeśli mimo wszystko potrzebujemy użyć tablicy ciągów, można je stworzyć samodzielnie poprzez skonfigurowanie pojedynczego "dużego" łańcucha, a następnie wywołaniu szeregu obliczeń w celu konwersji odniesienia do wiersza i kolumny na poprawne odniesienie do fragmentu ciągu. Oczywiście w pamięci komputera nie ma "wierszy i kolumn".

Dialekty Basica udostępniające tablice robią to poprzez symulowanie określonych pozycji, korzystając z listy adresów pamięci. Możemy łatwo robić to samo, symulując wiersze i kolumny korzystając z listy pozycji znaków w jednym, dużym łańcuchu. Jak to zrobić praktycznie? Załóżmy, że chcemy mieć tablicę ciągów składających się z 4 wierszy i 3 kolumn, przy czym każdy łańcuch w tablicy ma mieć maksymalną długość 20 znaków. Zaczynamy od ustawienia tych wielkości w zmiennych:

```
100 ROWMX=4 : COLMX=3 :
    LNGMX=20
```

Biorąc pod uwagę te wartości, wiemy, jak duża będzie nasza "tablica":

```
150 TTSIZ=ROWMX * COLMX *
    LNGMX
200 DIM ARR$(TTSIZ)
```

Obliczenia konwersji punktu odniesienia możemy teoretycznie wykonywać za każdym razem, gdy jest to potrzebne w programie. Jednak każde powtórzenie danego odniesienia oznaczałoby powtórzenie dokładnie tych samych czynności, więc szybciej i wygodniej obliczyć wszystko raz.

Później wyniki wystarczy przechowywać w taki sposób, aby były łatwo dostępne. Potrzebujemy jedną tablicę dla początkowych lokalizacji ciągów i jedną dla końcowych lokalizacji. Przykład podaję w ramce na następnej stronie (Program nr 1). Dzięki temu każde wystąpienie wyrażenia ARR$(X,Y) w Basicu może zostać zastąpione odpowiednikiem w poniższej postaci:

```
ARA$(BG(X,Y),EN(X,Y))
```

Rozwiązuje to część problemu związaną z konwersją adresów. W większości dialektów tablica może

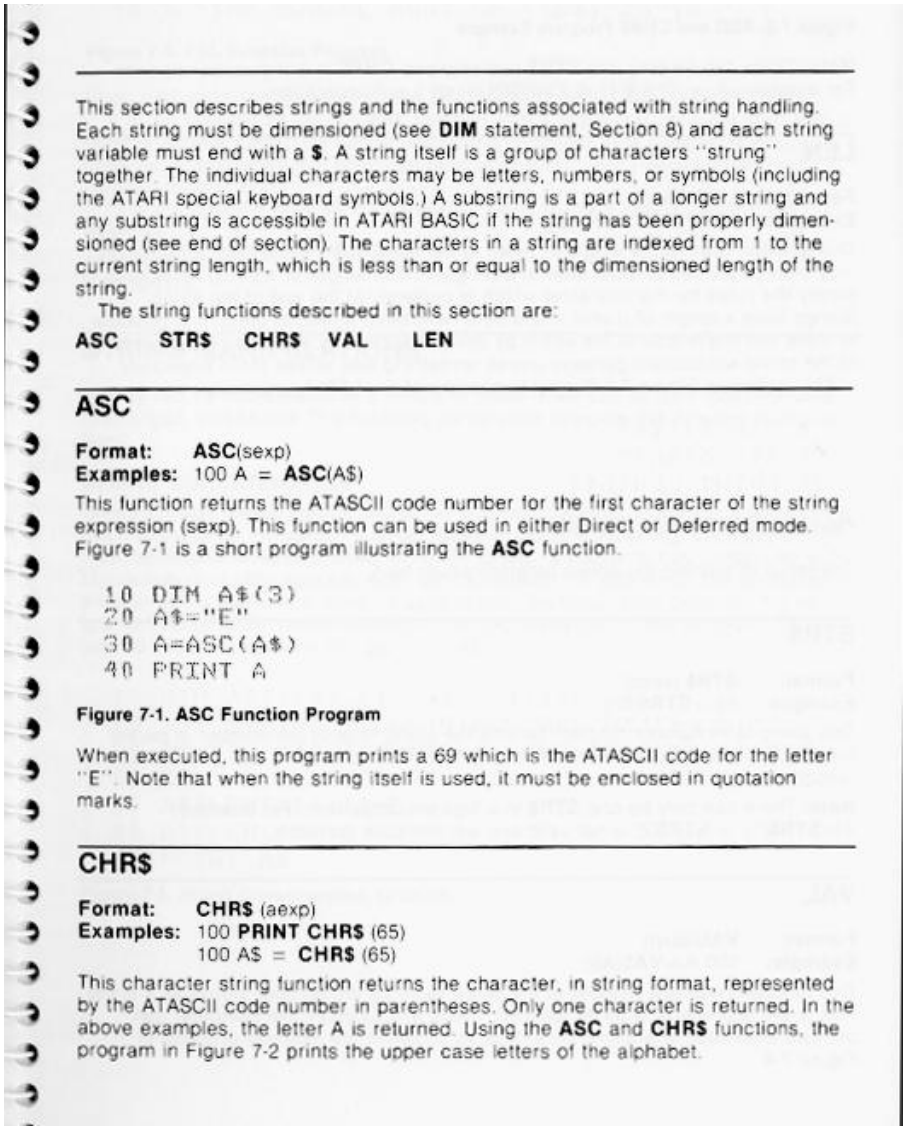
mieć długość równą zero lub dowolną inną, ale do pewnej wartości maksymalnej. W Atari Basic, wykorzystując naszą symulację, długość musi być dokładnie określona i wynosi ona tyle, ile zapiszemy w wartości LNGMX. Konsekwencje tego faktu zależą od zastosowania. Na przykład, tablica może testować pustą komórkę przy użyciu funkcji LEN:

```
IF LEN(A$(3,4))=0 THEN ...
```

Równoważny kod może obejmować ciąg o długości LNGMX zapisywany we wszystkich polach. Wtedy pusta komórka nie jest równa zero, lecz równa "pustemu" łańcuchowi, co można sprawdzić w ten sposób:

```
IF A$(BG(E,Y),EN(X,Y))=NUL$
THEN ...
```

Ponadto umieszczenie łańcucha krótszego niż LNGMX w symulowanej tablicy może wymagać uwzględnienia jej długości, tak jak poniżej:



This section describes strings and the functions associated with string handling. Each string must be dimensioned (see **DIM** statement, Section 8) and each string variable must end with a **$**. A string itself is a group of characters "strung" together. The individual characters may be letters, numbers, or symbols (including the ATARI special keyboard symbols.) A substring is a part of a longer string and any substring is accessible in ATARI BASIC if the string has been properly dimensioned (see end of section). The characters in a string are indexed from 1 to the current string length, which is less than or equal to the dimensioned length of the string.

The string functions described in this section are:

ASC STR$ CHR$ VAL LEN

ASC

Format: **ASC**(sexp)
Examples: 100 A = **ASC**(A$)

This function returns the ATASCII code number for the first character of the string expression (sexp). This function can be used in either Direct or Deferred mode. Figure 7-1 is a short program illustrating the **ASC** function.

```
10 DIM A$(3)
20 A$="E"
30 A=ASC(A$)
40 PRINT A
```

Figure 7-1. ASC Function Program

When executed, this program prints a 69 which is the ATASCII code for the letter "E". Note that when the string itself is used, it must be enclosed in quotation marks.

CHR$

Format: **CHR$** (aexp)
Examples: 100 **PRINT CHR$** (65)
100 A$ = **CHR$** (65)

This character string function returns the character, in string format, represented by the ATASCII code number in parentheses. Only one character is returned. In the above examples, the letter A is returned. Using the **ASC** and **CHR$** functions, the program in Figure 7-2 prints the upper case letters of the alphabet.

```
Program nr 1.

210 DIM BG(ROWMX,COLMX)
220 DIM EN(ROWMX,COLMX)
240 FOR RW=1 TO RDWMX : FOR CL=1 TO COLMX
250 BG(RW,CL)=COLMX * LNGMX * (RW-1) + (LNGMX * (CL-1) + 1)
260 EN(RW,CL)=BG(RW,CL)-1 + LNGMX
270 NEXT CL : NEXT RW
```

Oryginalna instrukcja Atari Basic omawia operacje na ciągach tekstowych, choć w dość uproszczonej formie.

```
ARR$(GB(X,Y),BG(X,Y)-1+LEN
(NEW$))=NEW$
```

Powyższa linia umieszcza zmienną NEW$ w komórce ARR$, zaczynając od pierwszej pozycji znaku komórki i zajmując tyle pozycji, ile wymaga długość NEW$. Linia ta jest oczywiście dłuższa i mniej zrozumiałą niż poniższa:

```
ARR$(X,Y)=NEW$
```

która mogłaby być użyta w dialektach Basica na innych komputerach. Ma jednak taką samą funkcję, pomijając nieco mniejszą prędkość działania. W przyszłych programach pewnym wyzwaniem może być optymalizacja tego algorytmu, co pozostawiam wytrwałym Czytelnikom naszego pisma.

Bufor wejścia/wyjścia

Opracował:
KAMIL STOKOWSKI

Transmisja danych na Atari nie należy do najszybszych. Dyskietka musi być fizycznie obracana, dlatego założenia DOS-u są takie, aby zminimalizować liczbę operacji. W związku z tym musimy poruszyć pojęcie tak zwanego bufora.

W systemie Atari DOS, jeśli odczytujesz lub zapisujesz dane do pliku, oznacza to, że wykonujesz operacje w 128-bajtowym obszarze zarezerwowanym w pamięci. To właśnie jest wspomniany bufor.

Dane w ogóle nie są zapisywane na dyskietkę, Atari tylko przechowuje zawartość jednego z sektorów w pamięci. Dlatego wszystkie operacje odbywają się z bardzo dużą prędkością, a tego właśnie chcemy. Oto przykład:

```
10 OPEN #1,4,0,"D1:FILE"
20 FOR A=1 TO 10000
30 GET #1,A
40 NEXT A
```

Ten krótki program odczytuje 10000 bajtów z dyskietki. Jeśli go uruchomisz, zauważysz, że nośnik nie jest stale dostępny — tylko raz na jakiś czas usłyszysz dźwięk wskazujący na kolejny odczyt. Atari otwiera plik i pobiera pierwszy sektor z danymi do bufora w pamięci.

Kiedy wykonujesz pierwsze 125 instrukcji GET, bajty są odczytywane z bufora. Następnie próbujemy odczytać kolejne bajty, ale Atari nie ma ich jeszcze w pamięci, stąd odczytuje kolejne 128 bajtów z dyskietki, zapełnia nimi bufor i ponownie zaczyna odczytywać dane. Skąd wzięła się liczba 125, zamiast 128?

Otóż Atari DOS rezerwuje trzy bajty na sektor do własnego użytku. Podobnie, gdy wykonasz instrukcję PUT, Atari pozwala na wysłanie 125 bajtów do bufora, następnie zapisuje je na dyskietkę, następnie przenosi kolejne 125 bajtow i tak dalej.

Rezultatem całego procesu jest fakt, że Atari nie musi używać nośnika podczas zapisywania każdego pojedynczego bajtu, a to przyspiesza operacje wejścia/wyjścia. Gdy używasz instrukcji LOAD lub SAVE, bufor używany jest ponownie. Ten procesor jest niewidoczny dla użytkownika, jednak zrozumienie tego mechanizmu jest ważne przy rozwiązywaniu niektórych problemów, które mogą wystąpić podczas korzystania z dyskietek.

Na przykład, jeśli zapisujesz dane w buforze, a program zawiesi się w międzyczasie, na dyskietce nie zostanie zapisana część informacji. W takich przypadkach należy użyć instrukcji CLOSE. Dzięki niemu możesz się upewnić, że wszystko, co znajdowało się w buforze, zostało również zapisane na nośniku.

Aktualizowany jest również katalog, w którym znajdują się wszystkie nazwy plików. Program DOS, obsługujący wszystkie operacje, utrzymuje tabelę wolnych sektorów na dysku. Kiedy bufor się zapełnia i potrzebne jest miejsce na przechowywanie kolejnych 128 bajtów, pobierany jest konkretny numer sektora.

Kiedy plik jest usuwany, jego sektory są zwracane do tabeli. W ramce przedstawiam bardziej skomplikowany przykład (Program nr 1). Po wydaniu polecenia COPY każdy odczytywany sektor jest wysyłany do pamięci. Innymi słowy, DOS kopiuje zawartość tego pliku do pamięci, po 128 bajtów na raz. Następnie pobiera plik z pamięci i zapisuje go na dyskietce — z pamięci, ponownie w częściach po 128 bajtów.

Aby wiedzieć, jakie sektory są używane i jakie pliki znajdują się na dysku, na dyskietce muszą być przechowywane specjalne tabele. Jedną z nich jest VTOC (Volume Table of Contents) i znajduje się w sektorze o numerze 360. To tutaj zapisywana jest tabela wszystkich 720 sektorów na dyskietce, przechowywana w formie tak zwanej mapy bitowej.

Katalog dysku jest przechowywany w sektorach o numerach od 361 do 368 i dotyczy to aktualnej listy nazw plików oraz sektorów, które zajmują.

Aby znaleźć konkretny plik, DOS musi użyć tego katalogu. Wszystkie powiązania są realizowane na podstawie tabeli zwanej Device Control Block. Dane są do niej wprowadzane, a następnie następuje skok do systemu operacyjnego, który korzysta z tabeli.

Aby odczytać dane z pliku, należy przejść do numeru sektora startowego znajdującego się w katalogu. Sektory mają numery od 1 do 720. Do określania kolejnych sektorów używany jest wskaźnik, który jest aktualizowany podczas odczytywania sektorów. W ten sposób pliki nie muszą być zapisywane w konkretnej kolejności na dysku — potrzebne sektory mogą być rozmieszczone na całym dysku.

Odczytywanie poszczególnych sektorów po kolei poprzez pobieranie kolejnych numerów jest nazywane "śledzeniem sektorów". Atari DOS robi to za każdym razem, gdy odczytuje dane z pliku. To samo możemy osiągnąć w prostym programu w Basicu. Należy odnaleźć dany wpis w katalogu i odczytać go zaczynając od numeru sektora podanego przez katalog.

Program nr 1

```
  5 REM SAMPLE PROGRAM TO DO DIRECT
  6 REM DISK I/O. DAVID SMALL,12/21/81
  7 REM
 10 DCB=768:REM START OF DISK DC3
 20 POKE DCB+1,1:REM DISK 1
 30 POKE DCB+2,82:REM $52 = GET SECTOR
 35 REM *** GET 128 BYTE OPEN BUFFER
 40 DIM BUFFER$(128)
 45 DIM CALL$(10)
 50 FOR X=1 TO 128:BUFFER$(X)=" ":NEXT X
 60 ADDR=ADR(BUFFER$)
 70 ADDRHI=INT(ADDR/256)
 80 ADDRLO=ADDR (ADDRHI*256)
 90 POKE DCB+4,ADDRLO:REM BUFFER ADR LOW
100 POKE DCB+5,ADDRHI:REM BUFFER ADR HI

110 REM *** SECTOR NUMBER ***
120 PRINT "INPUT SECTOR NUMBER"
130 INPUT SECTOR
140 SECTORHI=INT(SECTOR/256)
150 SECTORLO=SECTOR (SECTORHI*256)
160 POKE DCB+10,SECTORLO
170 POKE DCB+11,SECTORHI

200 REM *** BUILD SHORT ASSY PROGRAM
210 REM *** OF PLA, JMP DSKINV.
230 CALL$(1)=CHR$(104):REM PLA
240 CALL$(2)=CHR$(32):REM JSR
250 CALL$(3)=CHR$(83):REM $53
260 CALL$(4)=CHR$(228):REM $E4
270 CALL$(5)=CHR$(96):REM RTS
300 X=USR(ADR(CALL$))
310 DSTATS=PEEK(DCB+3)
320 PRINT "DISK STATUS=";DSTATS;" (1=OK)"
330 PRINT "DATA:"
340 PRINT BUFFER$
341 REM *** CLEAN OUT BUFFER$
342 FOR X=1 TO 128:BUFFER$(X)=" ":NEXT X
350 GOTO 110
400 END
```

Dalej przechodzimy przez każdy sektor, po uzyskaniu kolejnego numeru, aż do momentu odczytania wszystkich sektorów określonych w katalogu.

Operację taką wykonuje Program nr 2. Odczytuje katalog, wyświetla go na ekranie i pyta, który plik chcesz sprawdzić. Jeśli chcesz sprawdzić wszystkie pliki, wpisz "A". Jest to doskonały sposób, aby sprawdzić dyskietkę, na której mogą znajdować się pliki niemożliwe do odczytania.

Na koniec program tworzy listę pokazującą pliki odczytane poprawnie, uszkodzone oraz jeszcze niesprawdzone. Mając tę wiedzę, dość łatwo można napisać program kopiujący sektory.

To tylko kwestia odczytania wszystkich 720 sektorów, a później ich zapisania. Jednak niektóre dyskietki mogą być zabezpieczone przed taką operacją. Jak to działa?

Załóżmy, że mamy specjalną dyskietkę zawierającą kilka sektorów, które nigdy nie zostały sformatowane. Większość z nich jest w porządku, ale kilka jest uszkodzonych.

Gdy odczytujesz dane z tych sektorów, nie można ich zlokalizować właśnie dlatego, że nie były formatowane. W związku z tym uzyskiwany jest ERROR 144. Program sprawdza, czy błąd rzeczywiście wystąpił i kontynuuje swoje działanie.

W przeciwnym razie zostaje zatrzymany. Jedynym sposobem na ominięcie tego problemu jest usunięcie procedury kontroli sektorów w naszym programie. Oczywiście, w każdym przypadku może być wykonana kopia dyskietki, ale bez uszkodzonych sektorów program nie będzie działał.

Atari DOS jest potężnym narzędziem i pozwala na stosowanie wielu ciekawych technik. Dzięki nim Twoje programy mogą działać szybciej także pod względem transmisji danych, a to może być bardzo istotne, jeśli operujesz na większej ilości plików..

Program nr 2

```
 10 REM DISK CHECKER
 20 REM
 30 GOTO 290
 40 REM SUBROUTINES PLACED IN FRONT
 50 REM FOR SPEED. NO COMMENTING.
 60 REM
 70 REM DISK READ SUBROUTINE
 80 POKE DCB+1,DFROM
 90 POKE DCB+2,82
100 BHI=INT(BUFF/256):BLO=BUFF-(BHI*256)
110 POKE DCB+4,BLO
120 POKE DCB+5,BHI
130 SHI=INT(RSECTOR/256):SLO=RSECTOR-(SHI*256)
140 POKE DCB+10,SLO:POKE DCB+11,SHI
150 X=USR(CALL)
160 DSTATS=PEEK(DCB+3)
170 RETURN
180 REM DISK WRITE SUBROUTINE-(SAMPLE)
190 POKE DCB+1,BLO
200 POKE DCB+2,87
210 BHI=INT(BUFF/256):BLO=BUFF-(BHI*256)
220 POKE DCB+4,BLO
230 POKE DCB+5,BHI
240 SHI=INT(RSECTOR/256):SLO=RSECTOR-(SHI*256)
250 POKE DCB+10,SLO:POKE DCB+11,SHI
260 PRINT "ERROR":GRAPHICS 0
270 DSTATS=PEEK(DCB+3)
280 RETURN
290 REM DISK CHECKER.
300 REM DOES SECTOR TRACE OF ALL
310 REM FILES IN THE DIRTECTORY
320 REM PRODUCES LISTING WITH OK/
330 REM BAD.
340 REM ASSUMES 48K, MICROSOFT.
350 REM
360 REM GET DISK NUMBER
370 GRAPHICS 0
380 CLEAR
390 PRINT "WELCOME TO SMALL'S DISK CHECKER."
400 PRINT "INPUT DRIVE NUMBER";
410 INPUT DF$
420 IF DF$="" THEN DFROM=1:GOTO 440
430 DFROM=VAL(DF$)
440 DTO=DFROM
450 CALL=&E453   !DSKINV ADDRESS
460 REM NOTE: MICROSOFT CAN CALL
470 REM DSKINV$ DIRECTLY, NO PLA
480 REM NEEDED.
490 DCB=&300
500 OPTION RESERVE 20*128
510 B=VARPTR(RESERVE)
520 IF B<0 THEN B=B+65536
530 REM FIRST, READ IN DIRECTORY
540 PRINT "GETTING DIRECTORY."
550 INCR=0
560 FOR RSECTOR =360 TO 368
570 BUFF=B+(128*INCR)  !INPUT AREA
580 GOSUB 70           !GET SECTOR
590 IF DSTATS=1 THEN 630
600 PRINT "***DISK RETRY, DIRECTORY SECTOR
    ";RSECTOR
610 PRINT "***STATUS=";DSTATS
520 GOTO 570
630 INCR=INCR+1
640 PRINT INCR;
650 NEXT RSECTOR
660 PRINT
670 PRINT "WAIT -- BUILDING TABLES."
680 REM OK, NOW EXAMINE DIRECTORY.
690 REM PRODUCE LISTING FIRST OFF.
700 REM NOTE 360=VTOC, NOT DIR.
710 DIM INFO(64,6) !FILE INFO
720 DIM NIME(64) !NAME INFO
730 DIM DEAD(64) !FILE BOMBED?
740 FOR Z%=1 TO 64:DEAD(Z%)=0:NEXT Z%
750 REM
760 Z=B+128 !START OF DIR ENTRY
770 REM BUILD FLAG & INFO TABLE
```

```
 780 FNUM=1
 790 FOR LOC=Z TO Z+(64*16) STEP 16
 800 REM LOC=ENTRIES' STARTING ADDR
 810 FLAG%=PEEK(LOC)
 820 IF(FLAG% AND &80)=&80 THEN INFO(FNUM,3)=1
 830 IF(FLAG% AND &40)=&40 THEN INFO(FNUM,4)=1
 840 IF(FLAG% AND &20)=&20 THEN INFO(FNUM,5)=1
 850 IF(FLAG% AND 1)=1 THEN INFO(FNUM,6)=1
 860 REM SECTOR INFO
 870 INFO(FNUM,1)=PEEK(LOC+1)+256*PEEK(LOC+2)
 880 INFO(FNUM,2)=PEEK(LOC+3)+256*PEEK(LOC+4)
 890 REM NOW GET NAME DATA.
 900 NIME$(FNUM)=""
 910 FOR Z%=5 TO 15
 920 NIME$(FNUM)=NIME$(FNUM)+CHR$(PEEK(LOC+Z%))
 930 NEXT Z%
 940 REM CHECK FOR END OF ENTRIES
 950 IF PEEK(LOC+5)=0 THEN 1020
 960 REM
 970 FNUM=FNUM+1
 980 IF FNUM/5=INT(FNUM/5) THEN PRINT FNUM;
 990 NEXT LOC
1000 PRINT "DIR TOTALLY FULL."
1010 GOTO 1030
1020 PRINT "DIR PARTLY FULL."
1030 FNUM=FNUM-1
1040 PRINT
1050 PRINT "NUMBER OF ENTRIES=";FNUM
1060 REM OUTPUT DIRECTORY
1070 REM ENTRY POINT (FROM BELOW)
1080 IF FNUM=0 THEN 1320 !EMPTY DISK
1090 PRINT
1100 PRINT "D=DELETED/U=USED/L=LOCKED/O=OPEN"
1110 PRINT "SC=SECTOR COUNT(LEN)"
1120 PRINT "SS=STARTING SECTOR(LEN)"
1130 PRINT " + = GOOD FILE / - = BAD FILE / ? =
     UNKNOWN"
1140 PRINT
1150 FOR F=1 TO FNUM
1160 PRINT USING "##";F;
1170 IF DEAD(F)=-1 THEN PRINT "+";
1180 IF DEAD(F)=1 THEN PRINT "-";
1190 IF DEAD(F)=0 THEN PRINT "?";
1200 PRINT "!";NIME$(F);"!";
1210 IF INFO(F,3)=1 THEN PRINT "D"; ELSE PRINT
     " ";
1220 IF INFO(F,4)=1 THEN PRINT "U"; ELSE PRINT
     " ";
1230 IF INFO(F,5)=1 THEN PRINT "L"; ELSE PRINT
     " ";
1240 IF INFO(F,6)=1 THEN PRINT "O"; ELSE PRINT
     " ";
1250 PRINT "!SC:";
1260 PRINT USING "####";INFO(F,1);
1270 PRINT "!SS:";
1280 PRINT USING "####";INFO(F,2)
1290 IF F/20=INT(F/20) THEN INPUT RET$
1300 NEXT F
1310 GOTO 1390
1320 PRINT "EMPTY DISK -- NO ENTRIES"
1330 STOP
1340 PRINT
1350 REM ENTRY POINT (BELOW)
1360 PRINT "PRESS RETURN."
1370 INPUT RET$:CLS:GOTO 1420
1380 GOTO 1400
1390 PRINT "PRESS RETURN."
1400 INPUT RET$
1410 REM NEXT IS ENTRY POINT
1420 PRINT "SELECT :"
1430 PRINT "1. A FILE NUMBER TO CHECK;"
1440 PRINT "2. 'A' TO CHECK ALL FILES;"
1450 PRINT "3. 'D' TO SEE DIRECTORY LIST AGAIN"
1460 PRINT "NOTE: ERRORS SHOW UP ON DIR LIS-
     TING "
1470 PRINT "AFTER THEY OCCUR IN SEARCH."
1480 INPUT N$
1490 IF N$="A" THEN FILE=128:GOTO 1690
1500 IF N$"D" THEN GRAPHICS 0:GOTO 1070
1510 ON ERROR 1540
1520 FILE=VAL(N$)
1530 GOTO 1560
1540 PRINT "ERROR. RE-ENTER."
1550 RESUME 1420
1560 REM --- PROCESS. CHECK.
1570 IFFILE>FNUM OR FILE<0 OR FILE<>INT(FILE)
THEN PRINT "ERROR.":GOTO 1420
1580 REM SINGLE ENTRY HANDLER
1590 PRINT "TRACING=='SELECT' TO ABORT."
1600 SS=INFO(FILE,2) ! STARTING SECT
1610 SC=INFO(FILE,1) ! SECTOR COUNT
1620 RSECTOR=SS
1630 REM GO TRACE IT ...ARROR=RETURN
1640 GOSUB 1850
1650 IF ARROR=1 THEN PRINT "***BAD FILE***"
1660 IF ARROR=1 THEN DEAD(FILE)=1 ELSE
     DEAD(FILE)=-1
1670 GOSUB 2110 ! MARK FILE NAME
1680 GOTO 1350
1690 REM ALL ENTRIES HANDLER
1700 FOR FILE=1 TO FNUM
1710 CLS
1720 PRINT "TRACING FILE #";FILE
1730 IF LP=1 THEN PRINT #3,"TRACING FILE
     #";FILE
1740 PRINT "PRESS 'SELECT' TO ABORT."
1750 SS=INFO(FILE,2) ! STARTING SECT
1760 SC=INFO(FILE,1) ! SECTOR COUNT
```

```
1770 RSECTOR=55
1780 GOSUB 1850
1790 IF ARROR=1 THEN PRINT "***BAD FILE***"
     **"
1800 IF ARROR=1 THEN DEAD(FILE)=1 ELSE
DEAD(FILE)=-1
1810 GOSUB 2110
1820 NEXT FILE
1830 PRINT "COMPLETED."
1840 GOTO 1350
1850 REM TRACING SUBROUTINE
1860 ARROR=0
1870 PRINT
1880 PRINT "---SEC CNT=";SC;" READING SEC
     ";RSECTOR
1890 POKE &D01F,0 ' CLEAR BUTTONS
1900 GOSUB 70 'READ
1910 REM CHECK BUTTONS
1920 IF PEEK(&D01F)<>7 THEN RETURN
1930 IF DSTATS=1 THEN 1960
1940 PRINT "***ERROR. DSTATS=;DSTATS;" ON
     SECTOR ";RSECTOR
1950 GOTO 1900 !RETRY
1960 IF SC=0 THEN RETURN
1970 SC=SC-1 ! DECR S COUNT
1980 REM GET FORWARD POINTER + FNUM
1990 FWD=(PEEK(BUFF+125) AND 3)*256
2000 FWD=FWD+PEEK(BUFF+126)
2010 FN=INT((PEEK(BUFF+125) AND &FC)/4) !
WENCE, SWIFTED 2 BITS RIGHT
2020 FN=FN+1 ! (DISK STARTS AT 0)
2030 PRINT "---DATA: FILE NUM=";FN;" FWD SEC
     =";FWD
2040 IF FN<>FILE THEN PRINT "FILE NUMBER
     ERROR.":ARROR=1
2050 IF FWD>720 OR FWD<0 THEN PRINT "FORWARD
     PTR OUT OF RANGE,ABORT.:ARROR=1:RETURN
2060 IF FWD=0 THEN 2090
2070 RSECTOR=FWD
2080 GOTO 1870
2090 PRINT "COMPLETED."
2100 RETURN
2110 REM MARK A FILE AS BAD IN NAME.
2120 REM SET HIGH BIT SO IT SHOWS
2130 REM UP IN INVERSE VIDEO..
2140 IF ARROR=0 THEN RETURN
2150 Z$=NIME$(FILE)
2160 NIME$(FILE)=""
2170 FOR Z=1 TO LEN(Z$)
2180 NIME$(FILE)=NIME$(FILE)+CHR$((ASC
     (MID$(Z$,Z,1)) OR &80)) !SET HIGH BIT
2190 NEXT Z
2200 RETURN
```

Tester joysticka

Przedstawiona w ramce krótka procedura pozwala na łatwe testowanie joysticków i analizę potencjalnych problemów. Pomysł przyszedł mi go głowy, gdy na moim biurku pojawiło się kilkanaście joysticków i stanąłem przed wyzwaniem ich sprawdzenia oraz ewentualnej naprawy. Jest to bardzo prosty program, ale spełnia swoje zadanie. Po naciśnięciu przycisku Fire ekran robi się czerwony, natomiast kierunki oznaczane są kolorem niebieskim. Aby wyczyścić ekran, należy nacisnąć dowolny klawisz.

```
  2 GRAPHICS 3 : POKE 752, 1 :
    GOTO 999
  5 A = 24 : B = 14 : GOTO 200
  6 A = 24 : B = 6 : GOTO 200
  7 A = 25 : B = 10 : GOTO 200
  9 A = 16 : B = 14 : GOTO 200
 10 A = 16 : B = 6 : GOTO 200
 11 A = 15 : B = 10 : GOTO 200
 13 A = 20 : B = 15 : GOTO 200
 14 A = 20 : B = 5 : GOTO 200
 15 A = 20 : B = 10

200 IF C< >A OR D< >B THEN COLOR 4 :
    DRAWTO 20, 10 : COLOR 2 : DRAWTO
    A, B
230 COLOR 3 : PLOT 20, 10 : PLOT A,B
300 POKE 712, 66 - STRIG(0) * 66

500 POKE 656, 1 : POKE 657, 5
510 ? "STICK(0) = "; STICK(0),
520 ? "STRIG(0) = "; STRIG(0)
540 IF PEEK(764)< >255 THEN POKE
    764, 255 : RUN

999 C = A : D = B : GOTO STICK(0)
```

Atari Basic kontra PET

MARIUSZ WASILEWSKI

Przy zakupie komputera retro dla mnie liczą się nie tylko gry czy dema, ale także dostępne języki programowania. Wiadomo, że większość osób kieruje się sentymentem, ale jednak jeśli chcemy programować, warto poznać lepiej różnice pomiędzy dialektami wbudowanego Basica. Musimy zdecydować, jakie funkcje są ważne dla naszych przyszłych programów. W tym artykule przedstawiam podstawowe problemy, z jakimi trzeba będzie się zmierzyć.

Zacznijmy od zmiennych. Basic w komputerze PET pozwala tworzyć długie nazwy, ale minimalnie muszą być to dwa znaki. Mogą być one zakończone znakiem dolara ($) i tak jak zwykle, oznacza to wtedy zmienną tekstową. W Atari Basic nazwy mogą mieć dowolną długość, ale muszą zaczynać się od litery. W jednym programie można utworzyć 128 różnych zmiennych. Możliwe jest także tworzenie tablic, przy czym PET jest ograniczony do 256 elementów.

Ciągi tekstowe na Commodore mogą zawierać do 255 znaków, nawet jeśli bufor wejściowy jest ograniczony do 80 znaków. Do tworzenia dłuższych łańcuchów można używać operatorów. Na Atari nie ma ograniczenia długości ciągów, ale przy wprowadzaniu danych mamy limit do 99 znaków. Pętla na PET może być wykonana przy użyciu konstrukcji FOR...NEXT...STEP. Wartość kroku STEP może być liczbą całkowitą lub ułamkiem, liczbą dodatnią lub ujemną, co pozwala tym samym na wzrost lub spadek wartości indeksu. Niezależnie od tego, jakie są wartości początkowe i końcowe indeksu, pętla zostanie wykonana przynajmniej raz.

Pojedyncze polecenie NEXT może być użyte do zakończenia wielu poleceń FOR. Na przykład: NEXT X,Y,Z. W Atari Basic mamy te same możliwości, z wyjątkiem instrukcji NEXT, która nie może być używana dla wielu poleceń FOR.

Ponadto po NEXT musi następować nazwa zmiennej. PET pozwala natomiast na stosowanie samego słowa NEXT, wtedy zmienna wynika z rozpoczętej pętli w poprzedzających liniach programu. Polecenie INPUT na Commodore może zawierać komunikat zachęty, który zostanie

Klawiatura komputera Commodore PET model 2001 nie ułatwiała wprowadzania programów.

wyświetlony użytkownikowi przed znakiem zapytania. Polecenie to może być używane dla wszystkich typów zmiennych.

Na Atari nie może być dołączony komunikat, poza tym INPUT nie może być używany ze zmienną indeksowaną. PET pozwala na używaniu instrukcji READ w celu pobrania odpowiednich poleceń z linii DATA. Dane mogą być ponownie użyte, jeżeli użyjemy instrukcji RESTORE. Ten typ wprowadzania danych może być używany dla wszystkich typów zmiennych.

Taka sama sytuacja występuje w Atari Basic, gdzie konstrukcje READ...DATA również nie mogą być używane ze zmienną indeksowaną. W obu językach polecenie GET może być użyte do wprowadzenia pojedynczego bajtu, ale na Atari oczekiwane jest naciśnięcie klawisza. Podprogramy na PET są dostępne poprzez instrukcję GOSUB i potrzebują słowa RETURN, aby wskazać koniec procedury. Na Atari do zakończenia podprogramu możemy użyć instrukcji POP. Commodore posiada instrukcje warunkowe IF...THEN, które używają operatorów warunkowych takich jak >, <, >=, <=, <=, = oraz operatorów logicznych AND, OR, NOT do porównywania wartości zmiennych numerycznych i ciągów tekstowych.

Wiele instrukcji następujących po THEN zostanie wykonanych, jeśli warunek jest prawdziwy. W Atari Basic operatory AND, OR, i NOT nie działają na poziomie binarnym. W Basicu na komputerze PET polecenie PRINT może zawierać zmienne, wyrażenia arytmetyczne, ciągi znakowe i stałe. Z góry zdefiniowane

Ekran startowy komputera Commodore PET — informacja o wersji języka Basic oraz ilości wolnej pamięci..

są cztery domyślne pozycje na ekranie, które są używane, jeżeli argumenty są oddzielone przecinkami. Średnik pomiędzy nimi powoduje bliższe wypisanie wartości jako połączone ciągi znakowe oraz liczby oddzielone jednym znakiem Spacji.

Na Atari wygląda to bardzo podobnie, jednak średnik nie powoduje odstępów pomiędzy liczbami. Instrukcje STOP i END spowodują przerwanie wykonywania programu w przypadku PET, przy czym nie muszą być one zapisane jako ostatnie linie programu. Polecenie CONT pozwala na wznowienie wykonywania programu po napotkaniu słowa END lub STOP. W Atari Basic CONT powoduje kontynuowanie programu od następnej linii, ale niekoniecznie następującej po niej instrukcji.

Na Commodore dostępne są standardowe funkcje trygonometryczne i arytmetyczne, jak również funkcje specjalnego przeznaczenia, takie jak: PEEK czy TAB.

Ponadto funkcje mogą być zagnieżdżone. Na Atari dostępne są standardowe funkcje związane z ciągami, jak również funkcje specjalne typu PEEK. Dostęp do symboli graficznych na PET uzyskuje się poprzez naciśnięcie klawisza SHIFT i odpowiedniego innego klawisza. Symbole te mogą być używane w liniach z poleceniami PRINT do tworzenia grafiki na ekranie. Atari posiada również znaki specjalne i udostępnia specjalne słowa kluczowe ułatwiające tworzenie grafiki, takie jak PLOT, DRAWTO, POSITION, FILL i GRAPHICS.

Istnieje dziewięć różnych trybów graficznych: trzy tylko dla tekstu, dające normalne, podwójnie szerokie i podwójnie wysokie duże znaki, trzy

tryby z dzielonym ekranem i czterema kolorami, dwa z dzielonym ekranem i dwoma kolorami oraz jeden tryb wysokiej rozdzielczości. Jeśli chodzi o możliwości kolorystyczne, PET nie ma kolorów.

Na Atari przewidziane są specjalne słowa kluczowe służące do tworzenia kolorowych efektów, takie jak COLOR, który wybiera jeden z czterech rejestrów kolorów, oraz SETCOLOR do określenia barwy każdego rejestru. Przy użyciu kombinacji 16 odcieni i 18 ustawień jaskrawości można utworzyć aż 128 kolorów.

Na PET dźwięk uzyskuje się za pomocą instrukcji POKE. Atari posiada polecenie SOUND, które umożliwia określenie wysokości dźwięku, zniekształceń i głośności. Jednocześnie mogą być odtwarzane cztery głosy. Kontrola zniekształceń może prowadzić do tworzenia ciekawych efektów. PET nie ma żadnych specjalnych funkcji obsługujących sterowanie w grach. Atari Basic ma natomiast instrukcje ułatwiające programowanie obsługi joysticka. Są to: PADDLE oraz STICK. Na Commodore pliki muszą być otwierane przed użyciem wraz z parametrami określającymi numer pliku logicznego, numer urządzenia oraz nazwę pliku.

W przypadku taśmy nazwa może wynosić do 128 znaków, jednak system wykorzystuje tylko 16 znaków. Polecenie CLOSE służy do zamknięcia pliku i jako parametr wymaga tylko logicznego numeru pliku. Instrukcje PRINT#, INPUT#, GET# mogą być używane z magnetofonem lub plikami na dyskietce. Na Atari pliki muszą być otwierane wraz z parametrami określającymi numer pliku logicznego, rodzaj operacji (odczyt, zapis lub obie razem), nazwę pliku (8 lub mniej znaków) oraz typ urządzenia. Polecenia PRINT#, INPUT#, PUT#, GET#, i XIO mogą być używane podobnie jak na PET.

Oprócz standardowych poleceń NEW, LIST, RUN, CONT, LOAD, SAVE i POKE komputer PET posiada polecenie VERIFY umożliwiające weryfikację zapisu na taśmie. Instrukcje LOAD i SAVE mogą zawierać nazwę pliku. Polecenia na Atari są takie same jak PET, z wyjątkiem tego, że Atari nie posiada poleceń VERIFY lub CMD, a nazwy plików nie mogą być używane z poleceniami CLOAD i CSAVE.

Na Commodore podczas pisania programu możesz usunąć pojedyncze znaki lub cały wiersz. Późniejsza edycja jest możliwa poprzez sterowanie kursorem i usuwanie linii — po prostu wpisując numer linii i naciskając klawisz RETURN. Atari zachowuje się tak samo, jednak ma dodatkowo trzy funkcje edycyjne: wstawianie linii, usuwanie linii i funkcję Backspace.

W przypadku błędów składniowych PET podaje numer linii, ale często nie jest to przyczyna błędu. Na Atari tego typu błędy są sygnalizowane przez wyświetlenie linii oraz błędu w inwersie.

Jak widać, oba języki nie różnią się bardzo między sobą. Piszę to nie tylko po to, aby pokazać podobieństwa, ale chciałbym zachęcić posiadaczy obu sprzętów do przepisywania swoich programów na obie maszyny. Nie będzie to bardzo trudne, chyba że programy będą korzystały ze specyficznych funkcji dźwiękowych lub graficznych. Myślę jednak, że w erze powszechności emulatorów można sobie z tym poradzić względnie łatwo i szybko..

Przykładowy program w Atari Basic na ekranie telewizora kineskopowego.

Wykopaliska

Marble Madness

Opracował:
MARIUSZ WASILEWSKI

Marble Madness to gra wideo zręcznościowa zaprojektowana przez Marka Cerny'ego i wydana przez Atari w 1984 roku. Jest to gra zręcznościowa, w której gracz musi poprowadzić kulę przez sześć plansz wypełnionych przeszkodami i wrogami, a wszystko należy zrobić dodatkowo w ograniczonym czasie. Była to jednak z pierwszych gier w historii napisanych w języku programowania C, wykorzystująca dźwięk stereo.

Autor twierdził, że czerpał inspirację z mini-golfa oraz gier wyścigowych, a jego celem było stworzenie gry, która będzie miała unikalny system sterowania. Po wprowadzeniu na rynek, Marble Madness odniosła spory sukces. Krytycy chwalili poziom trudności gry, ciekawą oprawę wizualną i ścieżkę dźwiękową. Gra została przeniesiona na wiele innych platform niż Atari i z pewnością zainspirowała rozwój podobnych gier.

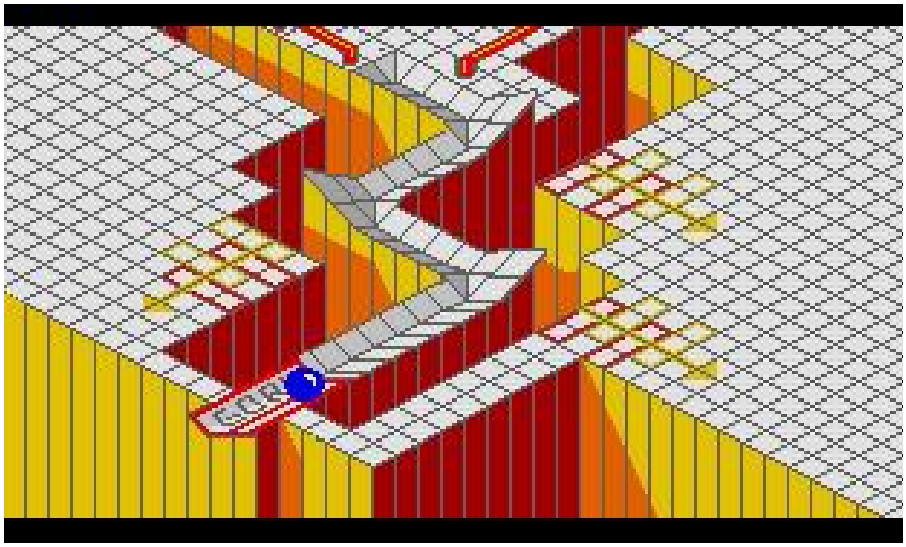
Celem gracza jest przeprowadzenie kuli przez sześć poziomów z widokiem izometrycznym, przypominających labirynt. Z wyjątkiem pierwszego etapu, czas pozostały na zegarze jest przenoszony do następnego poziomu, ponadto gracz otrzymuje dodatkowy czas na ukończenie gry.

Można też rywalizować ze sobą, grając w dwie osoby, w takiej sytuacji obaj gracze mają osobne zegary. Poziomy są wypełnione różnymi obiektami i wrogami zaprojektowanymi tak, aby jak najbardziej utrudnić grę. Kolejne poziomy stają się coraz trudniejsze i wprowadzają większą ilość przeszkód. Warto dodać, że każdy poziom ma inny styl wizualny.

Na przykład, pierwszy etap, pod tytułem "Practice", jest prostym torem, który jest znacznie krótszy od pozostałych, natomiast poziom piąty, nazwany "Silly", jest zorientowany w kierunku przeciwnym niż pozostałe plansze. Gra była pierwszą produkcją na Atari, w której wykorzystano układ dźwiękowy FM wyprodukowany przez Yamaha, który generował muzykę w czasie rzeczywistym tak, aby była zsynchronizowana z akcją na ekranie.

Ciekawostką jest fakt, że autor pracował nad grą bazującą na słynnym albumie Michaela Jacksona pod tytułem Thriller. Jednak projekt w pewnym momencie został odwołany i pierwotne pomysły ostatecznie zostały wykorzystane w Marble Madness. Gry na Atari były wcześniej programowane raczej w języku maszynowym.

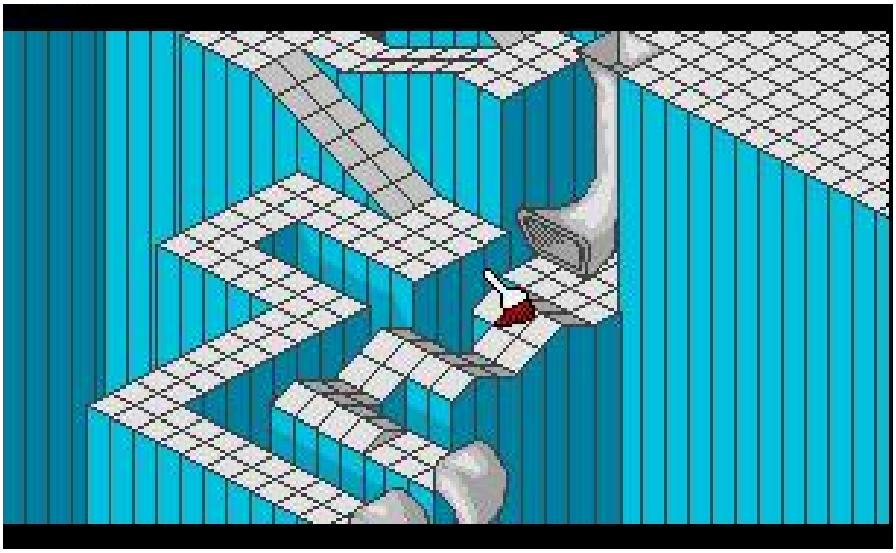
Język C był łatwiejszy do nauki, ale jednocześnie mniej wydajny, więc gra działa wolniej niż większość zręcznościówek z tego okresu. Ograniczenia zmusiły autora do uproszczenia projektów posz- czególnych poziomów. W przeciwieństwie do większości innych ówczesnych gier, grafika w Marble Madness nie jest rysowana na zasadzie "piksel po pikselu". Autor zapisał mapę wysokości poszczególnych punktów i testował układ za pomocą grafiki trójwymiarowej. Pozwoliło to stworzyć realistyczne cienie i dodać efekt antyaliasingu.



doczekała się wielu konwersji, więc z pewnością została udoskonalona od czasu premiery, jednak w wersji na Atari widać niewielkie niedociągnięcia.

Autorzy około roku 1991 chcieli też napisać drugą część Marble Madness, ale anulowali projekt ze względu na zbyt małe zainteresowanie ze strony wydawców. Wielka szkoda, bo rozwinięcie pierwotnego pomysłu mogłoby być prawdziwym hitem, nawet dzisiaj.

Gra ma zróżnicowane poziomy i nietypowe sterowanie, ale zachęca do rozgrywki wysoką jakością wykonania.



Przeciwnicy w grze musieli mieć niewielkie rozmiary ze względu na ograniczenia techniczne. Celowo pominięto twarze, aby nadać unikalny i minimalistyczny wygląd. Kierownictwo Atari zasugerowało jednak, że kula powinien mieć uśmiechniętą twarz, aby stworzyć rozpoznawalną postać, podobną do Pac-Mana.

Marble Madness w miarę rozwoju uzyskiwał coraz lepsze opinie wewnątrz Atari. Autor chciał dodać większą ilość poziomów i bardziej je rozbudować, niestety wydawca miał wówczas poważne problemy finansowe i nie zdecydowano się na wydłużenie okresu rozwoju gry. Z mojego punktu widzenia gra jest bardzo ciekawie napisana, ale jednocześnie na tyle trudna, że jeśli nie mamy większej ilości wolnego czasu — możemy się zniechęcić.

Nasza kula nie zawsze dobrze "przylega" do powierzchni, ponadto często bardzo trudno ocenić jej dokładną pozycję ze względu na widok izometryczny. Trzeba się przyzwyczaić do sterowania, ponadto kolizje z wrogami występują czasem w nieprzewidziany sposób. Gra

Kolejne lokacje przynoszą nie tylko bardziej skomplikowany labirynt, lecz także poruszających się przeciwników oraz "tajne przejścia".

MARBLE MADNESS

Nasza ocena:

grafika:

★★★★★★★★★★

dźwięk:

★★★★★☆☆☆☆☆

grywalność:

★★★★★★★★☆☆

Omawianą grę zawsze ceniłem bardzo wysoko, ale nie da się ukryć, że statyczna grafika w rzucie izometrycznym utrudnia rozgrywkę. Z drugiej strony ma to swój niepowtarzalny urok. Warto także dodać, że podobne gry powstały dla systemu Android, a więc praca autora Marble Madness nie została zapomniana.

Cartridge Assembler/Editor

Opracował:
ALEKSANDER KONDAŁA

Kiedy mówimy o asemblerze, zwykle mówimy o pakietach oprogramowania, które pozwalają nam pisać programy w języku maszynowym, a później je uruchomić. Takie pakiety zawierają trzy części: edytor, używany do pisania programów, asembler, używany do przekształcenia programu na język maszynowy, który faktycznie będzie działał, a także debugger, używany do wyszukiwania błędów.

Jednym z takich pakietów jest cartridge "Atari Assembler/Editor". Wymaga on, aby każda linia była poprzedzona numerem linii, tak jak każdy program napisany w Basicu. Numery linii muszą być liczbami całkowitymi w zakresie od 0 do 65535.

Po każdym numerze wiersza musi następować co najmniej jeden znak Spacji, na przykład:

```
10 LOOP LDA $0342 ;zaczynamy
```

Oprócz wspomnianych elementów mamy tu jeszcze komentarz umieszczony po znaku średnika, dzięki czemu wpis tekstowy jest ignorowany. Stosować możemy też etykiety, ale nie muszą być one obecne w programie.

Z reguły etykiety warto używać tylko wtedy, gdy wiemy, że później będziemy musieli odwoływać się do określonej linii z dalszej części programu. Pierwszy znak etykiety musi być literą od A do Z, a pozostałe znaki muszą być literami albo cyframi.

Etykieta może być krótka, nawet w formie jednego znaku lub dłuższa. Niektóre asemblery na Atari ograniczają liczbę znaków, które mogą być użyte w etykietach, dlatego najlepiej stosować nazwy nieprzekraczające 6 znaków.

Mnemonik, często nazywany kodem, jest instrukcją procesora 6502, którą chcemy, aby komputer wykonał. Przykładowo, jeśli chcemy, aby komputer załadował akumulator, służy do tego słowo LDA.

Instrukcja ta musi pojawić się albo z jedną pustą spacją między nią a etykietą, albo z dwoma pustymi spacjami między ostatnią cyfrą numeru linii a pierwszą literą mnemonika — jeśli nie ma etykiety (LABEL). Na przykład:

```
10 LABEL LDA $0342
```
lub
```
10 LDA $0342
```

Gdyby po numerze linii pozostało tylko jedno puste miejsce, asembler próbowałby zastosować mnemonik jako etykietę, czyli skończylibyśmy z etykietą o nazwie LDA. Asembler próbowałby wtedy zinterpretować operand ($0342) jako instrukcję procesora, co nie dałoby żadnego pozytywnego efektu.

Dla każdego mnemonika, który używa trybu adresowania akumulatora, operacja musi być zapisana wielką literą A, aby mogła być właściwie zinterpretowana. Dlatego, na przykład, instrukcja obracania zawartości akumulatora w prawo musi być napisana w ten sposób:

```
130 ROR A ; uwaga
```

Pole komentarza powinno opisywać opis wykonywanej operacji, ale

oczywiście nie jest konieczne wprowadzanie tekstów. Komentarze można jednak stosować nawet bez mnemonika, co sprawia, że kod jest bardziej czytelny. W tej sytuacji należy wpisać numer linii, a następnie odstęp, znak średnika i ponownie odstęp, na przykład:

```
100 ; Ta linia jest li-
      nia komentarza
```

Większość asemblerów ma do dyspozycji dla programisty szereg instrukcji, które mogą być interpretowane i rozszerzają zestaw instrukcji procesora 6502. Są to tzw. dyrektywy, które są one używane tak samo, jak mnemoniki, ale nie są częścią zestawu instrukcji procesora.

Jednym z najważniejszych jest określenie adresu pamięci. Asembler tworzy kod języka maszynowego, który będzie znajdować się w określonym miejscu w pamięci, dlatego musimy "powiedzieć" mu, jakie ma być miejsce docelowe. W przypadku omawianego cartridge'a należy to zrobić w następujący sposób:

```
10 *= $0600 ; poczatek
```

Zauważ, że pomiędzy ostatnią cyfrą numeru linii i gwiazdką znajdują się dwie spacje, natomiast nie ma spacji pomiędzy gwiazdką i znakiem równości.

Podobnie, jest tu jedna spacja pomiędzy znakiem równości i pierwszym znakiem adresu. Linia ta mówi asemblerowi, że chcemy, aby nasz kod rozpoczął składanie od szesnastkowego adresu $0600. Takie linie będą zazwyczaj pierwszymi lub jednymi z pierwszych w większości programów. Inne dyrektywy to między innymi:

.BYTE — rezerwuje co najmniej jedno miejsce w pamięci do przyszłego wykorzystania. Na przykład, instrukcja:

```
110 .BYTE 34
```

otwiera jedną lokalizację w pamięci przy aktualnej pozycji licznika programu i zapisuje w niej liczbę $22 (34 dziesiętne). Możliwe jest również zapisanie serii bajtów za pomocą jednej instrukcji BYTE, jak poniżej:

```
125 .BYTE "HELLO",$9B
```

.DBYTE — rezerwuje dwie lokalizacje dla każdej wartości w operandzie. Ta instrukcja jest używana dla danych, w których liczby są większe niż 256, a więc wymagają zapisania 2 bajtów. Numer jest zapisywany najpierw z bajtem górnym (bardziej znaczącym), a następnie bajtem dolnym (mniej znaczącym). Na przykład:

```
115 .DBYTE 300
```

.WORD — jest identyczny z dyrektywą .DBYTE, z tym wyjątkiem, że najpierw zapisywany jest bajt dolny, a następnie bajt górny. LABEL = jest używany do przypisania wartości etykiecie. Na przykład, jeśli napiszemy program, który wymaga częstego używania adresu $9F, możemy przypisać ten adres do nazwanej zmiennej w następujący sposób:

```
112 FREQ = $9F
```

Etykieta w dyrektywie LABEL musi zaczynać się od jednej pustej spacji pomiędzy ostatnią cyfrą numeru linii a pierwszym znakiem etykiety. Gdy później potrzebujemy adresu, możemy, zamiast niego odwołać się do etykiety:

```
245 LDA FREQ
```

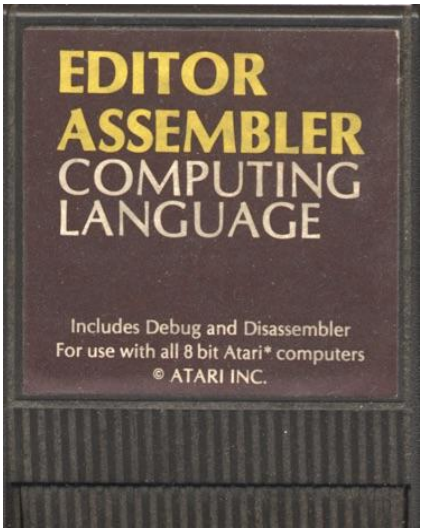
Teraz asembler "wie", jak załadować akumulator z adresu $9F.

.END — wskazuje, gdzie program powinien się zatrzymać. Oczywiście zwykle powinna to być ostatnia linia programu. Nasz cartridge zakłada, że jeśli nie ma kolejnych linii kodu, a jednocześnie nie ma dyrektywy .END, program jest zakończony. Dlatego dyrektywa .END jest opcjonalna, podobnie jak słowo END w programie w Atari Basic.

Istnieje wiele innych dyrektyw, ale powyższe są podstawowymi, jakie trzeba znać. Osoby zainteresowane tematem zapraszam do Biblioteki na stronie:

atarionline.pl

gdzie można znaleźć więcej materiałów na temat kodowania w Asemblerze Atari.



Gry z Agentem 007 w akcji

ADAM ZALEPA

Jamed Bond to postać znana od lat '50-tych XX wieku. Gry nawiązujące do Agenta Jej Królewskiej Mości powstają od początku lat '80-tych, gdy tylko komputery zagościły w naszych domach. Atari również otrzymało przynajmniej kilka takich gier. Dla mnie najważniejsze są dwa tytuły.

LICENCE TO KILL

Ta gra to oczywiście produkcja bazująca na filmie z agentem Jamesem Bondem. Trzeba powiedzieć, że wygląda całkiem nieźle. Grafika nie jest niczym specjalnym, ale jak na grę z 1989 roku jest solidnie wykonana. Pierwsze poziomy to czyta strzelanka ze scrollingiem z góry na dół — najpierw latamy w powietrzu, a później jeździmy na lądzie.

Porównując to z innymi tytułami na ST jak Ikari Warriors, Licence To Kill jest zdecydowanie bardziej przyjemna. Na niektórych poziomach występują co prawda spowolnienia, gdy na ekranie jest wiele akcji, ale nie są to bardzo częste sytuacje. Gra zawiera 6 scen z filmu, a po ich ukończeniu rozgrywka zaczyna się od nowa.

Ekran jest podzielony na dwie części. Większa część to obszar gry, a po prawej stronie widać Twój wynik i inne informacje, które zmieniają się w zależności od poziomu. W pierwszej scenie lecisz helikopterem. Po prawej stronie widzimy małą mapę i wysokościomierz.

Twoim zadaniem jest dotarcie do celu i zestrzelenie wrogich wieżyczek. Na mapie widać migającą kropkę, która reprezentuje Twój cel.

Druga scena to twoja pierwsza misja naziemna, jednak nie jest to tylko bieganie i strzelanie.

Trzeba ostrożnie celować i upewnić się, że jesteś zasłonięty przez jedną z wielu przeszkód, aby nie dać się trafić. Tym razem amunicja jest wyświetlana po prawej stronie ekranu. Kiedy zabijesz wroga, czasami otrzymujesz dodatkową amu-

nicję. Dalej gra staje się trudniejsza. Na następnym poziomie Twoim zadaniem jest zrzucenie Bonda z helikoptera na samolot — prawie tak samo, jak w filmie. Musisz dogonić samolot i ostrożnie lecieć tuż obok jego ogona.

W pewnym momencie pojawi się oznaczenie, dzięki czemu widać, kiedy i gdzie skierować naszego bohatera. Wszystko to musi być zrobione w ciągu około dwóch minut. Scena czwarta to poziom wodny. Po prawej stronie masz miernik tlenu. Musisz unikać łodzi motorowych i nurków z harpunami. Kiedy zanurzasz się, łodzie nie będą cię widzieć i nie będą strzelać, ale możesz szybko wyczerpać tlen. W tym trybie musisz uważać na innych nurków i próbować złapać harpun od wroga.

Od czasu do czasu na ekranie pojawi się samolot. Kiedy tak się stanie, Twoim zadaniem jest wycelować harpun w jeden z pontonów. Jeśli uda Ci się to zrobić, przejdziesz dalej. Będziesz musiał jeździć na nartach wodnych za samolotem i spróbować wspiąć się na niego, unikając skał i innych przeszkód. Szósta i ostatnia scena to bohater lecący w samolocie. Jesteśmy z powrotem w scenerii poziomu pierwszego, ale tym razem trzeba gonić ciężarówkę.

Ta część ma identyczną mechanikę jak w scenie trzeciej. Po prawej stronie widzimy limit czasowy i schemat ciężarówki przedstawiający poniesione przez Ciebie szkody. Gra jest bardzo nierówna pod względem grywalności. Wygląda bardzo dobrze, ale w niektórych momentach jest wręcz zbyt łatwa, a za chwilę...

LICENCE TO KILL

Nasza ocena:

grafika:

★★★★★★★★☆☆

dźwięk:

★★★★★★★☆☆☆

grywalność:

★★★★★★★☆☆☆

zbyt trudna. Poza tym nie mam większych zastrzeżeń i często wracam do Licence To Kill, nawet tylko po to, aby podziwiać naprawdę świetną grafikę.

LIVE AND LET DIE

Ostatnia gra, o której chcę powiedzieć, jest nietypowa, bo z jednej strony znowu nawiązuje do Agenta 007, a z drugiej posiada wiele ciekawych rozwiązań technicznych. Mamy tu między innymi potrójnie buforowany ekran używany dla uzyskania płynniejszej animacji, a także płynne skalowanie sprajtów, bo daje dużo lepszy efekt trójwymiarowości.

Gra rozpoczyna się na ekranie wyboru misji, gdzie możesz ustawić jeden z czterech poziomów. Pierwszy z nich to trening celowania, trzeba tu strzelać do czerwonych i czarnych celów. Mamy również dwie misje szkoleniowe. Każda misja jest rozgrywana na szlaku wodnym w innej części świata. Musisz kontrolować motorówkę za pomocą joysticka. Od samego początku można zauważyć, że — mimo teoretycznie prostej rozgrywki — w tej grze dzieje się bardzo wiele rzeczy. Droga wodna jest po prostu wypełniona przeszkodami, takimi jak skały czy miny.

Oprócz tego, mamy wrogie łodzie, wystrzeliwujące rakiety i samoloty na niebie. Łodzie wroga są łatwe do zniszczenia — po prostu użyj karabinu maszynowego. Samoloty to zupełnie inna sprawa. Będziesz musiał wypchnąć łódź w powietrze i strzelać do samolotów w bardzo krótkim czasie. Kiedy posuwasz się do przodu, kończy ci się paliwo — dzieje się to bardzo szybko.

Aby rozwiązać ten problem, można uzupełniać paliwo po drodze. Paczki z paliwem można też zniszczyć, ale wtedy stają się niebezpieczne także dla Ciebie. Dlatego naprawdę trzeba być bardzo ostrożnym i uważać, gdzie strzelamy.

Czasami amunicję i paliwo zrzuca helikopter, ale są to rzadkie przypadki. Moim zdaniem grę psuje trochę zbyt szybko kończące się paliwo. Nie raz będziesz musiał zbierać wszystkie paczki z paliwem, bo inaczej nie da się przejść poziomu. Z tego względu grać trzeba uważnie i nie można sobie pozwolić na zupełnie swobodne ruchy po ekranie. Mimo wszystko jest to gra, w którą warto zagrać i można się dobrze bawić.

LIVE AND LET DIE

Nasza ocena:

grafika:

★★★★★★★★★☆☆

dźwięk:

★★★★★★★☆☆☆

grywalność:

★★★★★☆☆☆☆☆

Dlaczego wciąż używam Basica?

Opracował:
KAMIL STOKOWSKI

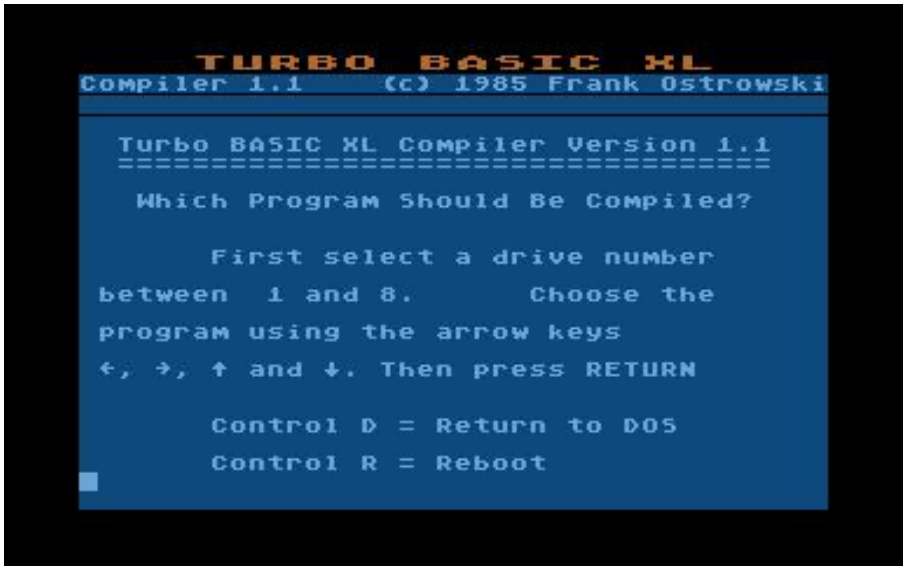
Osoby programujące w Basicu szybko odkrywają, że ich komputery mają specjalny "wewnętrzny" kod. Później, gdy dowiadują się więcej o asemblerze wydaje im się, że nie ma łatwego sposobu na nauczenie się kodowania.

W rzeczywistości programiści w asemblerze często uważają z kolei Basic za skomplikowany, źle zaprojektowany i obarczony niejasnymi regułami składni. Czy można znaleźć złoty środek?

Każdy język programowania ma swoje zalety i wady i moim zdaniem nie można powiedzieć, że któryś z nich jest lepszy. Basic jest szczególnie dobry do obliczeń w sytuacji, gdy nasz program musi być często zmieniany. Język maszynowy jest natomiast używany tam, gdzie liczy się szybkość działania.

Programiści w Basicu są zazwyczaj skoncentrowani na danych i obróbce informacji. Koderzy w języku maszynowym pracują an bazie mapy pamięci komputera. Moim zdaniem obie techniki mogą współdziałać i wzajemnie się uzupełniać także dzisiaj. Basic jest łatwiejszy do nauczenia. Mamy swobodę edycji programu i funkcje taki jak zmiana wiersza, wstawiania nowych instrukcji oraz sprawdzania programu — jest tu dużo ułatwień.

Ze względu na dużą ilość instrukcji program w Basicu można napisać szybko i od razu obserwować efekty jego pracy. Możemy korzystać z poleceń do wprowadzania i zapisywania informacji, a z funkcji takich jak zmienne czy tablice korzystamy w bardziej naturalny sposób. Osiągnięcie tego wszystkiego w asemblerze jest trudniejsze, bo trzeba je zaprogramować samodzielnie. Jed-



Kompilator Turbo Basica XL. Język jest zgodny z Atari Basic, lecz zawiera poprawki, a także potrafi działać nawet od 3 do 5 razy szybciej.

nak kod maszynowy działa bardzo szybko. Programy mogą uruchamiać się wielokrotnie szybciej niż ich odpowiedniki w Basicu.

Poza tym możemy uzyskać praktycznie bezpośredni dostęp do wewnętrznych mechanizmów działania, do których Basic nie może dotrzeć. Programy w Basicu można za to łatwo przenosić pomiędzy różnymi komputerami, co w przypadku asemblera jest zwykle niemożliwe lub wymaga przepisywania całych algorytmów. Dlatego uważam, że jeśli Basic może wykonać założone operacje w sposób wystarczająco szybki, nie ma sensu męczyć się z kodem maszynowym.

Dopiero w sytuacji, gdy nasze działania wykraczają poza możliwości Basica, należy pomyśleć o użyciu asemblera. Istnieją także inne powody, dla których dobrze jest znać język maszynowy. Daje on wgląd w wewnętrzne "bebechy" komputera. Trzeba zdać sobie sprawę, że przecież sam Basic jest tylko ogromnym programem języka maszynowego przechowywanym w pamięci ROM. Każda instrukcja w Basicu jest wykonywana przez dziesiątki małych instrukcji języka maszynowego.

Jeżeli chcemy wiedzieć dokładnie, jak działa polecenie w Basicu, trzeba prześledzić działanie kody maszynowego. Najlepiej pomyśleć o Basicu i asemblerze jako o uzupełniających się narzędziach. Basic może wywołać program w języku maszynowym, kiedy jest to potrzebne, za pomocą polecenia SYS lub funkcji USR. Kod języka maszynowego może zwrócić dostęp do Basica, gdy zakończy pracę używając instrukcji RTS. Dane mogą być przekazywane w obie strony pomiędzy tymi dwoma językami, a więc nie mamy ograniczeń pod względem wymiany informacji i ich dalszej obróbki lub konwersji.

Asembler nie wybacza błędów i w wielu wypadkach nie "powie", dlaczego program nie działa. Jego nauka może być zniechęcająca, dlatego osobiście wolę pisać programy w Basicu i tylko uzupełniać je kodem maszynowym tam, gdzie na to sens. Wydaje mi się, że ma to szczególne znaczenie dzisiaj, gdy wiele osób wraca do sprzętu retro i nie ma wiele czasu na programowanie.

Główną część programu można w ten sposób napisać szybciej i przy okazji mamy motywację, aby ukończyć projekt, który jest już prawie skończony. Nawet gdy to "prawie" jest najtrudniejszą częścią listingu.

Opcje edytora ekranowego dialektu Fast Basic.

Okno tekstowe w trybie GRAPHICS 0

Opracował:
MARCIN LIBICKI

Okno tekstowe może być użyteczną funkcją w trybach graficznych, umożliwiającą jednoczesne wyświetlanie tekstu i grafiki. Posiada bardzo podobne funkcje do zwykłego ekranu tekstowego. Obsługiwane są wszystkie funkcje edycyjne, a przewijanie i czyszczenie ekranu jest ograniczone do małego czteroliniowego okna.

Podobna sama funkcja byłaby przydatna w przypadku trybu GRAPHICS 0. Na przykład, menu z opcjami do wyboru może być wtedy prezentowane w około 20 górnych wierszach ekranu, a wprowadzanie danych przez użytkownika dokonywane w czterech dolnych wierszach okna tekstowego.

Wszelkie błędy, takie jak naciskanie klawiszy kursowa, zamiast wpisywania informacji w ramach instrukcji INPUT, nie będą kolidowały z resztą ekranu. Jak to wszystko zrobić? Bardzo łatwo — za pomocą tylko jednego polecenia POKE. Adres 703 zazwyczaj zawiera liczbę 24. Jeśli zapiszemy tu wartość 4, kursor znajdzie się w dolnej części ekranu i uzyskamy okno tekstowe.

Zauważ, że w tym przypadku nie możesz wyświetlać informacji w górnej części ekranu za pomocą instrukcji PRINT. W zamian musisz użyć PRINT#6 tak jak w przypadku trybów graficznych 1 i 2. Ponadto, polecenia POSITION dotyczą tylko górnej części ekranu, natomiast do pozycjonowania wyjścia okna tekstowego należy użyć kolejnych instrukcji POKE.

Pamiętaj, że dopóki nie użyjesz polecenia POKE 703,4, nie musisz też używać instrukcji PRINT#6. W ramce przedstawiam przykładowy program, który demonstruje użycie okna tekstowego.

```
100 REM
130 TRAP 150
140 OPEN #1,6,0,"D:*.*":GOTO 160
150 ? "Can't read directory":END
160 GRAPHICS 0:COL=0:POKE 752,1
170 DIM A$(20),F$(14):TRAP 230
180 INPUT #1;A$
190 POSITION COL,LINE:? A$(1,14)

200 LINE=LINE+1
210 IF LINE>20 THEN COL=COL+13:
    LINE=0
220 GOTO 180
230 POKE 703,4
240 FOR I=1 TO 100:? I,:NEXT I
250 ? CHR$(125):? "Run which
    program";:INPUT A$

260 TRAP 290
270 F$="D:":F$(3)=A$
280 RUN F$
290 ? "Can't RUN ";F$;"."
300 END
```

Prosty test szybkości

Opracował:
KAMIL STOKOWSKI

Szybkość wykonywania operacji matematycznych na "małym" Atari można przyspieszyć. Przykładem może być pakiet FASTCHIP, który zawiera własne procedury i według producenta przyspieszał komputer nawet do 3 razy. Instalacja polega na wymianie układu ROM na płycie głównej. Niezależnie od sposobu na zwiększenie szybkości działania, dobrze wiedzieć, w jaki sposób sprawdzić różnicę. Można uruchamiać różne programy, ale nie zawsze będzie to miarodajny test. Dlatego proponuję napisanie krótkiego programu testującego.

Trudno wyodrębnić ogólną zasadę, jaką przyjmują programy mające pokazać prędkość działania komputera. W zależności od algorytmu będzie to mniej lub bardziej odpowiadało faktycznej różnicy w pracy, na przykład podczas uruchamiania programów w Basicu.

W sieci znalazłem wiele różnych przykładów, także takich, które próbują symulować średnią prędkość działania. Wydanie mi się to jednak karkołomnym pomysłem.

Najprostszym sposobem jest wykonanie obliczenia zajmującego spory czas procesora. Program mmoże obliczać pierwszą liczbę pierwszą z zakresu wartości większych od miliona:

```
10 FOR I = 1000001 TO 1000003 STEP 2
20 FOR D = 3 to SQR (I) STEP 2
30 IF I/D = INT(I/D) THEN 60 40 NEXT D
50 PRINT I
60 NEXT I
```

Warto tu zwrócić uwagę, że nie wszystkie 8-bitowe komputery posiadają wystarczającą precyzję, aby wykonać podobne operacje. Atari powinien wykonać nasz program w kilkanaście sekund, co jest czasem nieco dłuższym niż na Apple II czy Commodore PET. Znaczący wzrost prędkości można osiągnąć poprzez wyłączenie układu ANTIC odpowiedzialnego na generowanie obrazu.

Porównanie wykonania testowego algorytmu na różnych komputerach:

TRS color Computer	37 sekund
Commodore PET	30 sekund
Commodore VIC-20	27 sekund
Apple II	26 sekund
TRS-80 model II	23 sekundy
Apple III	20 sekund
Sinclair ZX-81	13 sekund
Osborne I	8 sekund
Atari 400/800 (ANTIC ON)	15 sekund
Atari 400/800 (ANTIC OFF)	10 sekund
Atari 400/800 (ANTIC ON, FASTCHIP)	11 sekund
Atari 400/800 (ANTIC OFF, FASTCHIP)	8 sekund

Źródło: Compute! magazine

Należy pamiętać, że syntetyczny test polegający na wykonaniu jednej operacji nie daje realnego porównania różnic w prędkości działania różnych komputerów. Daje to jednak orientację, który dialekt języka Basic jest szybszy.

Podobną technikę stosuje się przykładowo na Commodore 64, bo przecież podczas tego typu operacji i tak czekamy na wynik i nie ma znaczenia, czy widzimy na ekranie nieruchome tło, czy tymczasowo wygasimy obraz. Efekt, o jakim wspominam, można osiągnąc używając instrukcji:

```
POKE 559,0
```

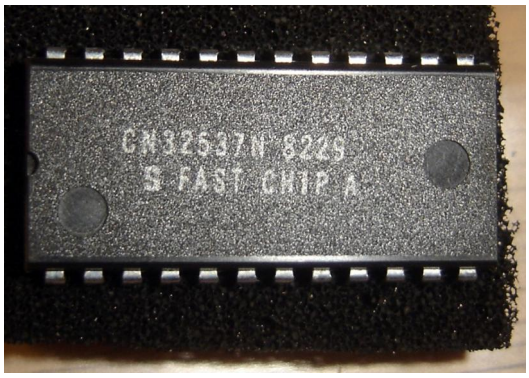
co spowoduje wyłączenie wyświetlania obrazu do momentu zakończenia naszego programu. Później należy użyć tej samej komórki pamięci, ale zapisać wartość 34, zamiast "zera".

Dzięki temu czas wykonania programu może skrócić się nawet o 30-40%, według moich testów może to być około 3 sekundy na 10 sekund pracy. Jest to więc duży wzrost wydajności. Kolejnym pomysłem ja sprawdzenie szybkości działania komputera może być obliczenie sumy tysięcy wartości funkcji kwadratowej.
Oto inny przykład:

```
 50 S=0 : X=0
200 FOR N = 1 TO 1000
300 S=S+X*X
400 X = X + 0.00123
500 NEXT N
600 PRINT S,X
```

Ze względu na wielokrotne dodawanie liczb zmiennoprzecin- kowych, program ten wykonuje przy okazji test precyzji obliczeń i wystepowania ewentualnych błędów, co ma miejsce w niejednym Basicu.

W Internecie znalazłem też porównanie prędkości niektórych popularnych maszyn, które przedstawiam w tabeli. Wynika z niej, że w niektórych przypadkach Atari działa wolniej nawet od ZX Spectrum, co mnie osobiście zdziwiło. Nie należy tu się sugerować szybkością procesora, lecz bardziej wydajnością zaimplementowanych algorytmów Basica.



W dziesiątym numerze naszego magazynu przedstawione były niektóre błędy wpływające także na prędkość pracy, tak więc osoby bliżej zainteresowane tym tematem zapraszam do lektury pozostałych numerów RetroKompa.

Bufor klawiatury

Opracował:
KAMIL STOKOWSKI

Atari nie posiada standardowo bufora klawiatury. Oznacza to, że naciśnięcia klawiszy w momencie, gdy komputer jest "zajęty" są tracone. Na szczęście można to zmienić za pomocą specjalnego programu. Będziemy korzystać z zarezerwowanego obszaru pamięci komputera do tymczasowego przechowywania serii naciśnięć klawiszy w czasie, gdy klawiatura jest nieaktywna. Wszystkie zapisane naciśnięcia klawiszy zostaną wypisane na ekranie dopiero w sytuacji, kiedy klawiatura będzie aktywna i znów będzie można wprowadzać informacje.

Na początek spójrz na ten prosty program:

```
10 GOTO 10
```

Kiedy go uruchomisz, komputer będzie wykonywał nieskończoną pętlę. Jeśli w tym czasie wpiszesz znaki, komputer zignoruje to. Używając bufora klawiatury nadal nic nie widzisz, gdy uruchamiasz program i wpisujesz znaki. Jednak, gdy tylko zatrzymasz program naciskając klawisz BREAK, wszystkie wpisane wcześniej znaki zostaną wyświetlone.

Taki efekt możesz uzyskać po wpisaniu programów zamieszczonych w ramkach. Jest on zaprojektowany do użycia jako podprogram w ramach innego programu w Basicu wymagającego wprowadzanie danych z klawiatury. Po jego wpisaniu zapisz go, a później będziesz mógł go połączyć ze swoim programem.

Jeśli naciśniesz RESET podczas korzystania z bufora, konieczne będzie ponowne uruchomienie programu poprzez wpisanie:

```
A = USR(1536)
```

Jeśli natomiast pojawi się błąd lub komunikat CHECK DATA STATEMENTS, oznacza to, że popełniłeś błąd podczas wpisywania programu. Należy wtedy dokładnie sprawdzić wszystkie linie DATA.

Zwróć uwagę, że linie od 30000 to główna procedura bufora, a linie od 20 do 110 to test jego działania. Gdy wszystko będzie w porządku, komputer wyświetli napis:

```
WHAT IS YOUR NAME?
```

i rozpocznie odtwarzanie dźwięków. Niezależnie od tego, że odtwarzana jest muzyka, wpisz swoje imię i naciśnij RETURN. Po zakończeniu odtwarzania Twój wpis zostanie wyświetlony na ekranie automatycznie. Jest to tylko jeden z przykładów zastosowania bufora klawiatury.

Jeśli chcesz używać bufora klawiatury podczas wykonywania własnego programowania w Basicu, zmień linię o numerze 30150 na następującą:

```
30150 END
```

a następnie uruchom program. Gdy pojawi się komunikat READY, wpisz:

```
NEW
```

Bufor klawiatury będzie działać i możesz rozpocząć pisanie programu.

Bufor klawiatury może poprawić jakość każdego programu wymagającego wprowadzenia danych przez użytkownika. Znaki można wprowadzać nawet w sytuacji, gdy komputer wykonuje długą pętlę.

Dzięki temu możesz uzyskać nietypowe efekty działania programu, a poza tym oszczędzić czas podczas wprowadzania większej ilości danych.

```
20 GOSUB 30000
30 DIM NAME$ (30)
40 PRINT "WHAT IS YOUR NAME?";
50 FOR I = 1 TO 500
60 A = RND(0) * 255
70 SOUND 1, A, 10, 8
80 NEXT I
90 INPUT NAME $
100 PRINT "YOUR NAME IS "; NAME$
110 END
30010 DATA 104, 173, 8, 2, 141, 96, 6, 173, 9, 2, 141, 97, 6, 169, 0, 141, 14, 212,
           120, 169, 52, 141, 8, 2
30020 DATA 169, 6, 141, 9, 2, 169, 98, 141, 36, 2, 169, 6, 141, 37, 2, 169, 192, 141,
           14, 212, 169, 0, 133, 204
30030 DATA 133, 205, 88, 96, 173, 9, 210, 201, 159, 240, 36, 152.72, 173, 252, 2,
           201, 255, 240, 19, 164, 204, 192, 100, 165, 204, 197, 205.208, 231, 104,
           168, 76
30050 DATA 95, 6, 173, 252, 2, 201, 255, 208, 35, 165, 204, 197, 205, 240, 23, 230,
           205, 164, 204, 192, 120, 176, 15, 164
30060 DATA 205, 192, 120, 176, 9, 185, 143, 6, 141, 252, 2, 76, 98, 228, 169, 0, 133,
           204, 133, 205, 76, 98, 228
30070 T = 0
30080 IF PEEK (521) = 6 THEN GOTO 30150
30090 FOR I = 1 TO 143
30100 READ N:T = T + N
30110 POKE 1535 + I, N
30120 NEXT I
30130 IF T <>18309 THEN PRINT "CHECK DATA STATEMENTS":STOP JA 30140 A = USR(1536)
30150 RETURN
```

Ulepszony RAM Dysk

Opracował:
MARIUSZ WASILEWSKI

Funkcja RAM Dysku dostępna w Atari XE pozwala uzyskać dużą szybkość i natychmiastowy dostęp do programów i plików. Jest to jedna z najciekawszych funkcji, jaką spotykamy najczęściej w sprzęcie 16-bitowym. Chcę przedstawić program, który przenosi wybrane pliki automatycznie do RAM Dysku, przy każdym uruchomieniu komputera. Jest to bardzo wygodne rozwiązanie, a wymagana jest tylko stacja dyskietek o Atari DOS 2.5.

Pliki po przeniesieniu do RAM Dysku są dostępne niemal natychmiast, ale kopiowanie każdego z nich ręcxznie jest żmudnym procesrem. Program, o którym mówię automatyzuje ten proces za pomocą niestandardowego pliku o nazwie AUTORUN.SYS. Po uruchomieniu komputera, automatycznie przesyła on wybrane programy w Basicu i pliki tekstowe z domyślnego dysku D1: na RAM Dysk, czyli D8:.

Aby cała procedura zadziałała poprawnie, komputer z systemem DOS 2.5 i przejdź do menu, aby wybrać opcję L. Następnie załaduj plik SETUP.COM. Użyj opcji 2, aby utworzyć plik AUTORUN.SYS. Dalej, wróć do Basica i wpisz program. Zwróć uwagę, że instrukcja DATA w linii 30 powinna zawierać nazwy programów w Basicu lub plików tekstowych, które chcesz przenieść na RAM Dysku po włączeniu zasilania.

Przy dodawaniu tych nazw należy podać pełną nazwę i rozszerzenie, ale bez symbol napędu. Innymi słowy nie należy umieszczać D1: na początku nazwy. Rozszerzenie nazwy musi mieć dokładnie trzy znaki długości. W razie potrzeby należy dodać dodatkowe spacje, aby uzyskać odpowiednią długość. Ostatnią pozycją DATA w tej serii musi być END, co zadziała jako oznaczenie końca listy nazw plików.

Kiedy wpiszesz linię 40, zamień nazwę programu jaki chcesz uruchomić podczas uruchamiania komputera. Na przykład, jeśli chcesz uruchomić MYPROG.BAS z napędu D1:, linia 40 powinna wyglądać tak:

```
40 READ F$:IF F$="END" THEN
   RUN "D1:MYPROG.BAS"
```

Należy zachować ostrożność podczas wpisywania linii 290 i 560, które zawierają drobne procedury języka maszynowego przechowywane w ciągach tekstowych. Muszą być one wpisane poprawnie, w przeciwnym razie komputer zawiesi się. Tego samego programu można użyć do przenoszenia programów z urządzenia D1: do D2: (zamiast do D8:), bez konieczności ręcznego kopiowania każdego pliku. Trzeba tylko wykonać drobne modyfikacje użytych symboli.

Drugą możliwością jest całkowite wyeliminowanie linii DATA i odczytanie nazw plików z wcześniej utworzonego pliku, a nie z linii zawierających instrukcje DATA. Za pomocą polecenia takiego jak INPUT#1,STRING$ można pobrać nazwę każdego pliku, który ma być skopiowany. Plik może zakończyć się też nazwą następnego uruchomionego programu. Wystarczy zastosować instrukcję warunkową jak poniżej:

```
IF STRING$="END" THEN IN-
PUT#1,STRING$:RUN STRING$
```

Program na początku odczytuje nazwy plików za pomocą instrukcji DATA w linii 30. Jeśli nazwa nie jest słowem END, program przeszukuje kolejne sektory. Następnie zmienna FLEN zachowuje swoją długość. Podprogram GETBYTES określa, czy jest to program w Basicu, czy też plik zawierający tekst lub inne dane. Nagłówek pliku programu zawsze

zaczyna się od dwóch zer, dlatego zakładamy, że wszystko, gdzie nie ma tych wartości nie jest programem w Basicu.

Kolejne sześć par bajtów nagłówka zawiera informacje o wielkości i lokalizacji niektórych wskaźników pamięci. Interesują nas ostatnie dwa bajty, które mówią ile jeszcze bajtów trzeba załadować, aby znaleźć koniec pliku (zmienna DEND).

Procedura IOCB służy do odczytywania bajtów ze tablicy FROM$ do ciągu ZZ$. Następnie ciąg jest modyfikowany w celu umożliwienia odczytu tekstu, a także innych danych lub programu w Basicu.

Przed zapisaniem ciągu, musimy znaleźć prawdziwy koniec danych. Plik tekstowy będzie miał mniejszą objętość niż FLEN*125 bajtów. Poprzez wyeliminowanie zerowych bajtów, otrzymujemy rzeczywistą długość pliku.

Podobny program może służyć do wielu zastosowań. Osobiście kopiuję automatycznie pliki, z których korzystam najczęściej lub takie, których wczytywanie zajmuje najwięcej czasu.

```
 15 POKE 712,148:POKE 559,0:POKE 8,255:POKE 731,1
 20 DIM A$(128),ZZ$(125*150),F$(15),FROM$(15),RDISK$(15),B$(16),FNAME$(16)
 25 TRAP 710
 30 DATA JUNK.1  ,JUNK.2  ,D1TOD8.BXE,END
 40 READ F$:IF F$="END" THEN RUN "D1:NEXTPROG.SAV"
 50 FOR SNUM=361 TO 368
 60 CLOSE #1:FLEN=0
 70 A$=CHR$(0):A$(128)=CHR$(0):A$(2)=A$
 90 DRIVE=1:TYPE=82:BUF=ADR(A$):GOSUB 260:REM "DISC" ROUTINE
100 GOSUB 330:REM "DECODE" ROUTINE
110 IF FLEN THEN SNUM=368
120 NEXT SNUM
130 REM
140 FROM$="D1:":FROM$(LEN(FROM$)+1)=F$:RDISK$=FROM$:RDISK$(2,2)="8"
150 GOSUB 600:REM "GETBYTES"
170 INDEX=BYTES*(BYTES<>0)+FLEN*125*(BYTES=0)
180 ZZ$="":ZZ$(1)=CHR$(0):ZZ$(INDEX)=CHR$(0):ZZ$(2)=ZZ$
190 OPEN #2,8,0,RDISK$:OPEN #1,4,0,FROM$
200 IOCB=1:TYPE=7:BUF=ADR(ZZ$):GOSUB 500:REM "IOCB" FOR READ
210 IF BYTES>0 THEN 220
211 IF ZZ$(LEN(ZZ$))=CHR$(0) THEN ZZ$=ZZ$(1,LEN(ZZ$)-1):GOTO 211
212 INDEX=LEN(ZZ$)
220 IOCB=2:TYPE=11:BUF=ADR(ZZ$):GOSUB 500:REM "IOCB" FOR WRITE
230 CLOSE #1:CLOSE #2
240 GOTO 40
250 END
260 REM PROCEDURE "DISC"
270 POKE 779,INT(SNUM/256):POKE 778,SNUM-256*INT(SNUM/256)
280 POKE 769,DRIVE:POKE 773,INT(BUF/256):POKE 772,BUF-256*INT(BUF/256):
    POKE 770,TYPE
290 X=USR(ADR("h 5d(.)")):REM D.104,32,83,228,96 or small h,
    space,Cap. S, inverse small d, ctrl-.
300 RETURN
310 REM TYPE=82 FOR READ, 87 FOR WRITE
320 REM
330 REM PROCEDURE "DECODE"
340 FLEN=0
350 FOR A=1 TO 8
360 B$=A$((A-1)*16+1,A*16):IF ASC(B$(1,1))>127 THEN GOTO 460
370 FLEN=ASC(B$(2))+256*ASC(B$(3))
380 FSTART=ASC(B$(4))+256*ASC(B$(5))
390 FNAME$=B$(6,13)
394 IF FNAME$(LEN(FNAME$))=" " THEN FNAME$=FNAME$(1,LEN(FNAME$)-1):GOTO 394
400 FNAME$(LEN(FNAME$)+1)=".":FNAME$(LEN(FNAME$)+1)=B$(14,16)
410 IF FNAME$=F$ THEN A=8:GOTO 470
440 FLEN=0
470 NEXT A
480 RETURN
490 REM
500 REM Procedure "IOCB"
510 REM ASSUMES IOCB ALREADY OPEN FOR READ OR WRITE
520 BLOCK=832+IOCB*16
530 POKE BLOCK+2,TYPE:REM READ=7,WRITE=11
540 POKE BLOCK+5,INT(BUF/256):POKE BLOCK+4,BUF-256*INT(BUF/256)
550 POKE BLOCK+9,INT(INDEX/256):POKE BLOCK+8,INDEX-256*INT(INDEX/256)
560 I=USR(ADR("hhh*LVd"),IOCB*16):REM h,h,h,inverse *, L,V, inverse d
570 CLOSE #IOCB
580 RETURN
590 REM
600 REM PROCEDURE "GETBYTES"
610 OPEN #1,4,0,FROM$
620 GET #1,I:GET #1,J
630 IF I<>0 OR J<>0 THEN BYTES=0:GOTO 690
640 FOR X=1 TO 6
650 GET #1,I:GET #1,J
660 NEXT X
670 DEND=256*J+I
680 BYTES=DEND-256+14
690 CLOSE #1
700 RETURN
710 REM
720 POKE 559,34
730 ? "ERROR ";PEEK(195);" AT LINE ";PEEK(186)+256*PEEK(187)
```

06.12 - 08.12.2019 (Friday-Sunday) / Gdańsk

SILLY VENTURE 2K19

Keeping The Legacy Of The Atari Scene Alive!

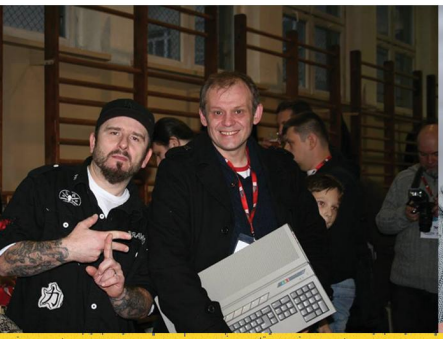
LYNX

30th ATARI LYNX ANNIVERSARY

11th DIMENSION OF TRUE ATARI PARTYING!

Największy i najbardziej niekonwencjonalny event dedykowany wszystkim fanom Atari na świecie!

Atari 2600, Atari XL/XE, Atari ST/STe/TT, Lynx, Falcon & Jaguar



Goście specjalni:
Wojciech Zientara ("Bajtek", "Moje Atari")
Jochen Hippel (Thalion Software)

więcej info na stronie **www.sillyventure.eu**