Polskie Pismo Amigowe

Numer 1/2012 (7)

cena 23 zł

Grafika: Sławek Woźniak

APEL O ARTYKUŁY!

Drodzy Czytelnicy,

Czekamy na Wasze artykuły, które zasilają bazę artykułów do kolejnych numerów. Przed nadesłaniem artykułu prosimy o skonsultowanie tematu z redakcją. Artykuły prosimy nadsyłać w postaci plików tekstowych (ASCII) wraz z dołączonymi obrazkami lub zdjęciami (format PNG lub JPG). Propozycje oraz sugestie należy nadsyłać na adres podany w stopce redakcyjnej.

Konkurs na artykuł

Redakcja PPA ogłasza konkurs na napisanie artykułu do *„Polskiego Pisma Amigowego”*. Temat artykułu powinien pasować do profilu pisma i powinien być wstępnie uzgodniony z redakcją (np. e-mailowo na adres kontaktowy *redakcja@ppa.pl*). Artykuł zgłoszony do konkursu nie może być krótką notką, orientacyjne minimum to 10 000 znaków oraz ilustracje, co pozwoli wypełnić dwie strony pisma.

Redakcja zastrzega sobie prawo do wydrukowania każdego ze zgłoszonych artykułów w *„Polskim Piśmie Amigowym”*. Artykuł wybrany do druku nie może być nigdzie opublikowany ani przed, ani rok po wydaniu go drukiem w PPA.

W miarę możliwości będziemy starać się opublikowane artykuły nagradzać albo w drodze plebiscytu na najlepszy tekst numeru lub w postaci wierszówek wypłacanych za zapełnioną stronę pisma.

Oprócz tego, tradycyjnie, każdy autor artykułu zakwalifikowanego do druku otrzyma bezpłatny egzemplarz pisma.

Termin nadsyłania artykułów upływa **30 kwietnia 2012** roku.

Zapraszamy do udziału!

Polskie Pismo Amigowe

Redaktor naczelny: Sebastian Rosa.

Zespół redakcyjny: Grzegorz Kraszewski, Grzegorz Murdzek, Piotr Sadowski, Krzysztof Żegleń, Konrad Czuba.

Korekta: Aleksander Piotr Chyliński.

Skład: Sebastian Rosa

Kontakt: redakcja@ppa.pl, http://ppa.pl.

Skład pisma wykonywany jest w programie OpenOffice 3. Druk na urządzeniach cyfrowych firmy Ricoh.

Polskie Pismo Amigowe jest wydawane w wolnym czasie członków redakcji i autorów. Redakcja nie gwarantuje regularnego ukazywania się kolejnych numerów. Cena jaką płacisz za pismo pokrywa jedynie koszty jego wydawania, pismo nie przynosi zysków.

Poglądy wyrażane w artykułach są poglądami ich autorów i niekoniecznie odpowiadają stanowisku redakcji.

http://www.ppa.pl

Rozwiązanie konkursów

W odpowiedzi na nasz konkurs wpłynęły artykuły, które pozwoliły zapełnić numer, który właśnie trzymacie w rękach. W odróżnieniu od poprzedniej praktyki zdecydowaliśmy się docenić trud każdego z autorów za wkład w powstanie magazynu. Dzięki osobie, która chciałaby pozostać anonimowa, za każdą opublikowaną w magazynie stronę przewidzieliśmy tzw. „wierszówki". Na chwilę obecną może jest to skromna kwota, lecz niech to będzie pewnego rodzaju sposób na wyrażenie wdzięczności. Być może w przyszłości darczyńców będzie więcej i autorzy za swoją publikację będą nagradzani nieco hojniej. Dziękujemy za nadesłane artykuły, a darczyńcy za ofiarowany datek.

W numerze

Od redakcji

Nie ma co ukrywać. Ten numer naszego magazynu ledwo co powstał. Zasadniczo planowaliśmy utrzymać cykl czterech numerów w roku i na zeszłoroczne święta byliśmy w 90% gotowi, lecz jak zwykle zabrakło artykułów. Pozostało kilka stron pustych i nie było zbytnich nadziei na szybką zmianę tego stanu rzeczy. Do tego doszły jeszcze inne zawirowania, które miały wpływ na tak znaczące opóźnienie i przesunięcie wydania numeru o prawie dwa miesiące (na pewno rzucił Wam się w oczy brak komiksu). Niemniej jednak w końcu udało się i mamy przyjemność przekazać w Wasze ręce siódmy numer naszej gazetki.

Numer zdominowany jest przez tematykę skierowaną do użytkowników Amig klasycznych i to zarówno tych mniej, jak i bardziej doświadczonych. Na początek obszerny artykuł porównujący sloty Zorro oraz PCI. Dowiecie się z niego, co kryje się pod tymi nazwami oraz co każda z nich ma do zaoferowania potencjalnemu amigowcowi stojącemu przed dylematem zakupu. Następny w kolejności jest artykuł o wadach i zaletach dysków twardych w porównaniu z kartami pamięci. Tutaj podobnie – osoby wahające się z całą pewnością znajdą w nim odpowiedź na kilka dręczących je pytań. Na zakończenie sprzętowych rozterek „klasykowców" mamy artykuł z cyklu retro – recenzja Amigi CD32 – dziecka firmy Commodore, które miało stanąć w szranki z oponentami na rynku konsol. Jest to pierwsza część obszernego artykułu. Na ciąg dalszy zapraszamy do kolejnego numeru, w którym autorzy postanowili przyjrzeć się zapleczu rozrywkowemu dostępnemu na tę platformę.

Skoro już przeszedłem na tematy rozrywkowe, nie można nie wspomnieć, że siódmy numer naszego pisma skierowany jest również do graczy. Znajdą oni w nim dla siebie aż cztery teksty. Prezentujemy w nich recenzję *„Mr Beanbag"* – nowej gry platformowej rozwijanej od parunastu ładnych lat, a której nie powstydzili by się wydawcy z lat świetności Amigi, recenzję przeportowanej gry *„Free Heroes2"* oraz *„Legion"*, która została przepisana pod bibliotekę SDL, a także artykuł skierowany do wszystkich fanów gier przygodowych Sierry, w którym autor skupił się na opisaniu wszystkich (dostępnych na Amigę) części sagi *„King's Quest"*.

W numerze nie zabrakło również czegoś dla lubiących coś więcej niż składać komputery i grać w gry. Kontynuujemy kurs programowania pod AmigaOS oraz prezentujemy artykuł–zachętę na temat pisania własnych skryptów w Directory Opus Magellanie. Temat bardzo na czasie, zwłaszcza że w ostatnich dniach deweloper oryginału zgodził się uwolnić kod źródłowy programu za „drobną opłatą". Na zakończenie mamy jeszcze recenzję książki *„Bajty polskie"*, którą otrzymaliśmy od autorów w ramach nagrody w konkursach, za co im serdecznie dziękujemy.

Jak wspomniałem na wstępie, numer wyraźnie skierowany jest do użytkowników Amig klasycznych. Zdajemy sobie sprawę, że może w ten sposób zostać ograniczona liczba odbiorców, lecz nie można zapominać, że pismo tworzone jest przez Was – naszych czytelników. To wy budujecie jego zawartość. Jeżeli chcecie więcej czytać o nowościach, o tematyce związanej z AmigaOS 4, MorphOS-em czy AROS-em, to musicie o tym napisać. W przeciwnym razie na nic zda się lament, że pismo Wam tematycznie nie odpowiada. Na moment zamknięcia tego numeru nie mamy rezerwy tekstowej na numer ósmy, a więc trudno przewidzieć kiedy znowu będziecie mogli trzymać pisemko w rękach. Liczę na to, że teksty wkrótce napłyną i spotkamy się już za kwartał. Jeżeli miałoby to nie nastąpić... wówczas pozostaje mieć nadzieję, że następny raz jednak kiedyś nastąpi...

Sebastian Rosa

YAM 2.7 SimpleMail 0.38

Pojawiła się nowa wersja YAM-a – programu pocztowego. Frank Weber, Jens Langner i Thore Boeckelmann przez ostatni rok pracowali głównie nad lepszą wydajnością i poprawianiem błędów. Znalazło się również miejsce na kilka nowości, takich jak: możliwość cytowania zaznaczonego fragmentu tekstu, zapamiętywanie szerokości kolumny okna preselekcji, rozpoznawanie niekompatybilnych formatów plików wiadomości podczas importu, całkowite usunięcie funkcji umożliwiającej wysyłanie anonimowych wiadomości, udostępnienie folderu SPAM jako domyślnie włączonego, możliwość ustawienia sposobu przekazania wiadomości (w tekście lub jako załącznik), możliwość uruchamiania skryptów po starcie programu lub po zakończeniu operacji filtrowania wiadomości, możliwość zaznaczenia więcej niż jednego pliku do importu wiadomości, możliwość ustawienia funkcji preselekcji dla każdego konta osobno, automatyczna zmiana filtrów w sytuacji zmiany nazw folderów, nowe algorytmy przeszukiwania wiadomości. Na koniec najważniejszy element – program działa wielowątkowo i jest to duży krok w kierunku nadal nieobsługiwanego protokołu IMAP.

http://www.yam.ch/

Zgodnie z tradycją, jak co roku w Wigilię, upubliczniona została nowa wersja programu pocztowego SimpleMail. Co nowego? Zaimplementowano obejście, które powinno wyeliminować prawdopodobne zawieszania związania z połączeniem SSL IMAP pod AmigaOS 4.x, usunięto problemy IMAP związane z logowaniem i przesyłaniem haseł zawierających spacje, dodano nowy tryb „Complete mails" dla folderów IMAP (wiadomości są pobierane w tle – a nie na życzenie), nowy system monitorowania postępu.

http://simplemail.sourceforge.net/

AmiKit 1.6.0

Po ponad dwóch latach ciszy pojawiła się aktualizacja pakietu AmiKit – prekonfigurowanej dystrybucji systemu AmigaOS przeznaczonej dla Windowsa, Macintosha i Linuxa. Niebywałą nowością jest rozpowszechnianie dystrybucji na pamięci USB (darmowy obraz do pobrania dostępny będzie w późniejszym terminie). Ponadto w jej skład weszły również pełne wersje programów AmIRC, FroggerNG, FryingPan oraz Directory Opus Magellna II, nowy silnik emulacji (WinUAE w wersji 2.4.0), zaktualizowane wersje oprogramowania (ScummVM, OpenSSL, NetSurf, WHDLoad, AfaOS, MCP, Scalos). Więcej informacji oraz szczegóły dotyczące kupna znaleźć można na stronie projektu.

http://amikit.amiga.sk/

Pierwszy kontakt

Firma A-EON Technology wyprodukowała pierwszą limitowaną serię „First Contact" komputerów AmigaOne X1000 przeznaczoną dla użytkowników końcowych. Komputer w podstawowej konfiguracji ma być wyposażony w:

- system AmigaOS 4.1 update 5 (z licencją na bezpłatną aktualizację do wersji 4.2, gdy tylko będzie dostępna),
- oficjalną obudowę z logo Boing Ball (w kolorze białym lub czarnym),
- 1 GB pamięci RAM,
- kartę graficzną Radeon 4650,
- dysk twardy 500 GB,
- nagrywarkę DVD,
- kartę muzyczną i sieciową.

Cena takiego zestawu (nie wliczając kosztów licencji systemu, wysyłki i podatku) ma wynosić 1699 funtów brytyjskich. Wkrótce w sklepie Amigakit.com będzie można składać zamówienia i dokonywać przedpłat na komputer (istnieje możliwość dokupienia dodatkowej pamięci, dysku twardego, napędu optycznego, myszy, klawiatury oraz takiego skonfigurowania komputera, aby działał również pod kontrolą systemu Debian Squeeze). Pierwsze egzemplarze udało się dostarczyć kupującym w styczniu 2012 roku.

http://www.a-eon.com/

OWB 1.16

Fabien Coeurjoly upublicznił wersję 1.16 opartej na silniku WebKit przeglądarki www Odyssey Web Browser dla systemu MorphOS. Archiwum z programem można pobrać ze strony autora. Zmiany obejmują uaktualnienie silnika do wersji r103170, możliwość pobierania plików większych niż 4 GB, dodanie obsługi drukowania w trybie Postscript oraz komendę drukowania przez port ARexxa, poprawienie funkcji eksportu do formatu PDF, dodanie opcji definiującej czy wyskakujące okienko powinno pojawić się w panelu, czy w oknie, możliwość wywołania strony, z której nastąpiło pobranie pliku, klawisz przełączenia w tryb pełnoekranowy działa również na obszarze wideo, uaktualnienie skryptu Youtube.js, a także poprawki błędów.

http://www.sand-labs.org/owb

AmigaOS.net

Wystartowała nowa strona internetowa poświęcona systemowi AmigaOS. Z jej zawartości łatwo wywnioskować, że autor kładzie nacisk na najnowszą odsłonę systemu i podkreśla istniejące oraz docelowe platformy sprzętowe jemu przeznaczone. Strona nadal jest w budowie – niektóre podstrony jeszcze nie istnieją. Autor jest otwarty na propozycje dotyczące tego, co powinno się tam znaleźć lub co należałoby zmienić. W planach jest między innymi stworzenie „skarbnicy wiedzy" o AmigaOS opartej na silniku Wiki.

http://www.amigaos.net/

AmigaOS 4.1 Update 4

Firma Hyperion Entertainment wydała czwarte uaktualnienie systemu AmigaOS 4.1. Wśród poprawek i zmian znalazły się:

- nowy folder „Emulation" z plikami ROM oraz plikami Workbencha w wersji dla AmigaOS 3.x z przeznaczeniem do emulacji starszych modeli Amig klasycznych oraz CD32,
- do systemu dołączone zostało RunInUAE,
- nowa wersja scsi.device przeznaczona dla użytkowników Amig klasycznych umożliwiająca wykorzystanie stronicowania pamięci (SWAP) na dyskach SCSI,
- dla użytkowników Amig klasycznych pojawiła się aplikacja NoDriveClick wyłączająca „klikanie" napędu dyskietek,
- obsługa DefIcons przez RunInUAE,
- uaktualnienie stosu TCP/IP o znacznie lepszą obsługę DHCP,
- poprawki MUI, Workbencha, obsługi USB,
- poprawka elf.library (programy korzystające z obiektów współdzielonych potrafiły dziwnie się zachowywać).

Uaktualnienie jest darmowe dla wszystkich zarejestrowanych użytkowników i dostępne jest dla każdej obsługiwanej platformy.

http://www.hyperion-entertainment.biz/

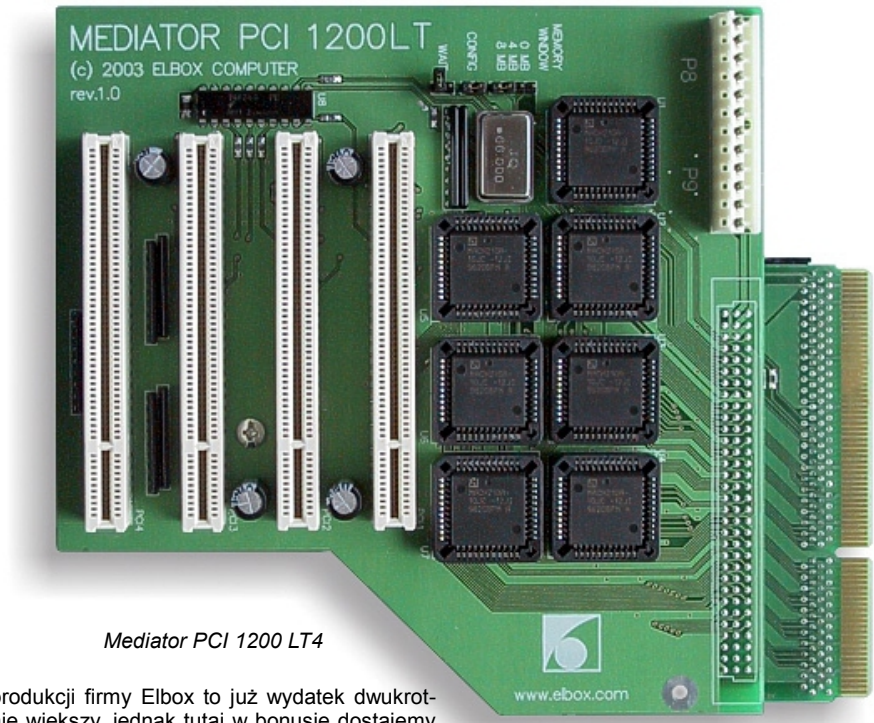
AmigaOS na netbooku

Podczas bankietu, który odbył się podczas AmiWest 2011, Steven Solie ogłosił iż system AmigaOS 4.x dostępny będzie na netbooku wyposażonym w procesor PowerPC. Dwunastocalowy sprzęt ma być oparty na architekturze e300, a jego dostępność zapowiadana jest na pierwszy kwartał 2012 roku. Zakładana cena (wraz z systemem) waha się od 300 do 500 dolarów. Wiadomość została również potwierdzona przez Bena Hermansa w wątku na forum AmigaWorld.net. Istnieje kilka prototypów maszyny wyposażonych w 512 MB pamięci i dwa porty USB, na których pracuje, w bardzo wczesnym stadium, system AmigaOS 4.1.

http://amigaworld.net/modules/newbb/viewtopic.php?topic_id=34459&forum=33

Zorro vs PCI

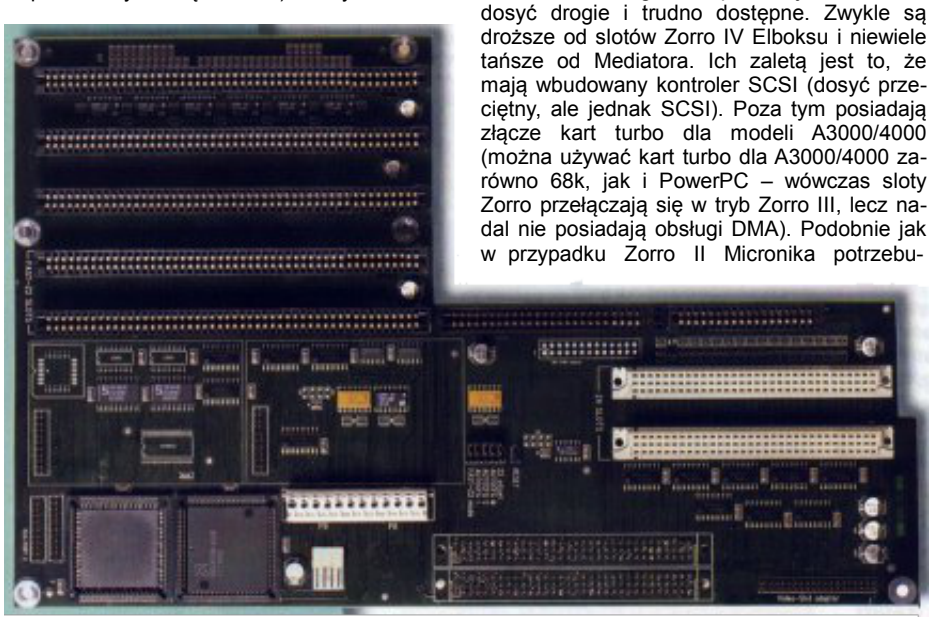
Kilkanaście lat temu amigowy świat zachłysnął się slotami PCI, które powoli zaczęły pojawiać się na amirynku. Pojawiły się takie rozwiązania PCI, jak Prometheus, Mediator, G-Rex. Każde z nich miało swoje wady i zalety, ale wszystkie dawały w większym lub mniejszym stopniu to, czego amigowcy nie mieli – dostęp do tanich i szybkich kart PCI. Najszybsza amigowa karta graficzna wpinana w klasyczne sloty Zorro nie dorastała do pięt dowolnej karcie wpinanej do slotu PCI. Podobnie było z dźwiękiem czy kartą sieciową. Za 30 ówczesnych złotych można było kupić całkiem niezłego Sound Blastera 128. Za drugie tyle amigowiec stawał się posiadaczem karty sieciowej na PCI. Tymczasem ceny odpowiadających im kart pod sloty Zorro przerażały (i nadal przerażają). Analogiczna karta dźwiękowa czy sieciowa to wydatek po około 400 zł za sztukę. Zmierzch slotów Zorro wydawał się przesądzony, lecz stało się inaczej. Pomimo ekspansji PCI w amigowy świat, karty Zorro II/III mają się całkiem nieźle i, co ciekawe, mimo archaicznych możliwości, powstają nowe karty przeznaczone dla tej magistrali. Przyjrzymy się plusom i minusom każdego z tych rozwiązań.



Mediator PCI 1200 LT4

Warto pamiętać, że o tym, z jakiego rozwiązania możemy skorzystać, będzie decydować posiadany przez nas model Amigi. Na przykład posiadacze A2000 nie będą mieć innego wyjścia niż stosować karty współpracujące wyłącznie z Zorro, a z kolei użytkownicy Amigi 3000/4000 będą mogli cieszyć się zarówno ze slotów PCI, jak i Zorro i używać ich równolegle. Bazą dla naszych rozważań będzie najpopularniejszy model – A1200.

Pierwszym elementem, na który zdecydowana większość z Was zwróci uwagę i na jego podstawie być może podejmie decyzję o zakupie, jest cena. Pozornie wygrywają sloty Zorro, które w chwili, gdy piszę te słowa można kupić już za około 150 zł (Zorro II Micronika). Do tego należy dokupić przelotkę zasilania (zbędna, jeśli posiadamy wieżę Infinitiv). Sloty Zorro IV produkcji firmy Elbox to już wydatek dwukrotnie większy, jednak tutaj w bonusie dostajemy cztery porty zegara. Nie trzeba również kupować przelotki, ponieważ te sloty mają standardowe złącze zasilania AT. Poza tym dwa pierwsze sloty są 32-bitowe i mogą działać znacznie szybciej niż oryginalne sloty Zorro II (kilka kart to wykorzystuje). Warto także nadmienić, że karta rozszerzeń Zorro II produkcji Elboksu ma na pokładzie dwa tajemnicze sloty ochrzczone mianem Zorro IV. Jest to patent Krakowiaków i podłączyć do tego możemy dwa urządzenia – Fast ATA oraz Mediatora dla slotów Zorro IV. W tym ostatnim przypadku możemy używać zarówno slotów PCI – jak i Zorro, jednak to rozwiązanie jest drogim przedsięwzięciem. Pozostają jeszcze sloty Zorro III dla Amigi 1200 produkcji Micronika – dosyć drogie i trudno dostępne. Zwykle są droższe od slotów Zorro IV Elboksu i niewiele tańsze od Mediatora. Ich zaletą jest to, że mają wbudowany kontroler SCSI (dosyć przeciętny, ale jednak SCSI). Poza tym posiadają złącze kart turbo dla modeli A3000/4000 (można używać kart turbo dla A3000/4000 zarówno 68k, jak i PowerPC – wówczas sloty Zorro przełączają się w tryb Zorro III, lecz nadal nie posiadają obsługi DMA). Podobnie jak w przypadku Zorro II Micronika potrzebujemy specjalnej przelotki zasilania, ponieważ Micronik w swoich slotach stosował własne rozwiązanie dotyczące złącza zasilania.



Zorro IV Busboard/Winner

Sloty Zorro są bezkonkurencyjne, jeśli chodzi o ilość kart, które możemy z nimi używać oraz o ilość wspieranych systemów amigowych i amigopochodnych. Większość urządzeń wpiętych w sloty Zorro bez problemów działa z AmigaOS 2.x/3.x, a część także z najnowszą odsłoną systemu – AmigaOS 4.x, jak również z systemem MorphOS. Podobnie jest z amigowymi dystrybucjami systemów Linux/Unix, które bez zająknięcia dogadują się z kartami pracującymi w slotach Zorro, czego niestety nie można powiedzieć o kartach wpiętych w mostki PCI (na razie żaden mostek PCI nie działa z Linuksem – z Uniksem działa wyłącznie Prometeusz).

Zostawmy na razie temat rozszerzeń Zorro i przejdźmy do PCI. Na rynku dostępne są trzy rozwiązania wspomniane na wstępie. Mediator PCI jest drogim rozwiązaniem, lecz, co dla niektórych może być istotne, nadal produkowanym. Zwykły model przeznaczony dla Amigi 1200 kosztuje około 500-550 zł. Jednak to dopiero początek kosztów. Najprostszą kartę graficzną, jaką możemy podłączyć do Mediatora, jest S3 Virge. Karta obecnie kosztuje kilka złotych. Jej odpowiednik dla slotów Zorro to CyberVision, za którą musimy zapłacić jakieś 500 zł. Obok Mediatora rozwiązaniem PCI dla Amigi 1200 jest mostek G-Rex. Aby zamontować go w Amidze, potrzebujemy karty Blizzard PPC. Mostek ten jest jednak trudno dostępny. Nie jest niestety obsługiwany przez AmigaOS 4.0/4.1, choć przez MorphOS 1.4.5 już tak. W porównaniu z Mediatorem wypada niestety słabo – mała ilość dostępnych sterowników, a także gorsze parametry, jeśli chodzi o szybkość czy busmastering. Wszystko to sprawia,

że G-Rex ustępuje miejsca Mediatorowi. To jednak nie jedyna przewaga „medka". Mediator powstał w naprawdę wielu wersjach. Właściwie każdy użytkownik znajdzie coś dla siebie. Dla samej Amigi 1200 mamy jego sześć wersji (ZIV, LT2, LT4, 1200, SX, TX). Różnią się one zarówno ilością aktywnych slotów PCI (od dwóch do sześciu), jak i szybkością działania i sposobem zasilania (AT/ATX). Poza tym jest wiele rożnych modeli przeznaczonych dla Amig 3000/4000. Niestety nie powstała wersja dla Amigi 2000. Mediator też jest najlepiej oprogramowanym mostkiem PCI dla AmigaOS i jako jedyny oferuje wsparcie dla USB przez kontroler USB Spider. Sytuacja Spidera była dosyć niejasna z powodu konfliktu na linii Elbox – Chris Hodges – twórcą stosu USB Poseidon, jednak całkiem niedawno pojawiła się informacja, że Poseidon znowu wspiera Spidera. Poza samym Mediatorem Spider do działania potrzebuje sterowników i wspomnianego już stosu. Co to oznacza w praktyce? Raczej możemy zapomnieć o używaniu klawiatury i myszki USB na przykład w Boot Menu (nawet do niego nie wejdziemy) czy też przy uruchomieniu systemu bez sekwencji startowej. W przypadku jakieś awarii nie zrobimy po prostu nic.

Mostek PCI Prometeusz został zaprojektowany przez Polaka – Grzegorza Kraszewskiego, a wyprodukowany przez firmę Matay na chińskich podzespołach. Tak więc osoba, która nabyła ten produkt, wspiera polską myśl technologiczną i chińską precyzję wykonania (nie mylić z jakością). Prometeusz jest mostkiem PCI wpinanym do slotów Zorro III. Aby go używać w Amidze 1200, potrzebne nam są sloty Zorro III Micronika oraz karta turbo dla Amigi 3000/4000 z zainstalowaną dodatkową pamięcią RAM. Jest więc to bardzo duży wydatek, biorąc pod uwagę fakt, że obecnie Prometeusz jest trudno dostępnym urządzeniem, a co za tym idzie – kosztownym.

Wydawać by się mogło, że wybór pomiędzy rozwiązaniem PCI a Zorro jest oczywisty i przynajmniej z dwóch względów – ekonomicznych i wydajnościowych – przemawia za tym pierwszym rozwiązaniem. Sloty Zorro mają jednak swojego asa schowanego nie w rękawie, ale w slocie. Pasjonaci–konstruktorzy opracowali kilka bardzo ciekawych rozwiązań, których na próżno szukać w świecie Mediatora i innych rozwiązań PCI dla Amigi. Doskonałym tego przykładem jest kontroler USB dla Zorro II – Deneb. Kosztuje on sporo, dużo więcej niż obsługiwane kontrolery USB dla Mediatora, ale posiada swój własny BIOS i na przykład myszka czy klawiatura podpięta do niego działa w Boot Menu, czego już Spider USB dla Mediatora nie potrafi. Mało tego, można podłączyć do Deneb kartę dźwiękową na USB, która kosztuje grosze, jak również wykorzystać ten kontroler do wykorzystania zwiększonej szybkości slotów Zorro IV produkcji Elboksu. Warto też dodać, że karta CyberVision wpięta w sloty Zorro potrafi przy niewielkim wysiłku wyświetlić obraz z Boot Menu bez użycia scandoublera, a sto razy tańszy S3 Virge dla PCI wpięty w Mediatora tej sztuki się niestety nie nauczył. Sloty Zorro posiadają również inne rodzaje złącz. Pierwszym jest slot video, który wykorzystywany jest najczęściej do podłączenia scandoublera, choć można również podłączyć genlock. Dzięki temu możemy do jednego monitora podłączyć zarówno sygnał PAL/NTSC z Amigi, jak i sygnał z karty graficznej. Co nam to daje w praktyce? Możemy nawet na mocno rozbudowanej Amidze z monitorem LCD cieszyć się grą w takie amosowe „hity" jak *„Rycerze mroku"*, *„Miasto śmierci"* i inne.

Nieco bardziej spostrzegawczy zauważą, że zarówno sloty Zorro II, jak i Zorro III Micronika mają na pokładzie sloty ISA, a niektóre modele również PCI. Niestety muszę ostudzić zapał – owe sloty są nieaktywne. Jedyne ścieżki, jakie do nich prowadzą, to ścieżki zasilające. Aby uaktywnić owe sloty, należy Amigę doposażyć w urządzenie zwane „Golden Gate" lub prawie dowolny sprzętowy emulator PC. Dzięki temu kilka typowych kart ISA, jak kontrolery IDE czy karty sieciowe, będą mogły pracować pod kontrolą AmigaOS 2.x/3.x. O kartach graficznych PCI/ISA używanych w tych złączach i pracujących pod kontrolą AmigaOS należy zapomnieć. Skoro o emulatorach mowa, to warto zaznaczyć, że powstało kilka kart, dzięki którym możemy sprzętowo udawać, poza PC, także Macintosha i Atari. Mało tego – nam amigowcom nieświadomie pomogła technologia przemysłowa i dzięki temu, że na rynku dostępne są komputery połówkowe, to do magistrali ISA/PCI, będącej częścią slotów Zorro, możemy podłączyć nowoczesny komputer PC i używać go równolegle z Amigą.

Zorro to również całkiem ciekawa magistrala dla wszelkich kontrolerów SCSI. Fajny bajer, gdy chcemy używać SCSI, a nie mamy kontrolera na naszej karcie rozszerzenia. Niestety jest to tylko bajer, bo Zorro jest zbyt wolne, aby efektywnie korzystać z podpiętych w ten sposób twardych dysków w ten sposób podpiętych. Lepiej użyć Fast ATA lub postarać się o kartę turbo z prawdziwym SCSI. Natomiast Mediator może obsłużyć zarówno kontroler SCSI, jak i SATA (pod AmigaOS 4.x), jednak nie uruchomimy naszej Amigi z dysku wpiętego do tego kontrolera. Najpierw musi zostać załadowany system i zainicjowane stosowne sterowniki. Z podobnymi ograniczeniami musimy się liczyć, gdy chcemy uruchomić jakąś

Prometheus

PCI (ang. Peripheral Component Interconnect) – magistrala komunikacyjna służąca do przyłączania kart rozszerzeń do płyty głównej. Po raz pierwszy została publicznie zaprezentowana w czerwcu 1992 roku jako rozwiązanie umożliwiające szybszą komunikację, od tej oferowanej przez standard ISA, pomiędzy procesorem i kartami rozszerzeń. Dodatkową zaletą PCI jest to, że każda karta, pasująca do gniazda PCI, funkcjonuje bez jakichkolwiek problemów, gdyż nie tylko sygnały, ale i przeznaczenie poszczególnych styków gniazda są znormalizowane. Szyna PCI stanowi kompleksowe rozwiązanie, przyspieszające współpracę z dowolnym urządzeniem zewnętrznym. Istotną cechą architektury PCI jest jej skalowalność: w jednym i tym samym komputerze może być równolegle lub szeregowo połączonych kilka magistral PCI. Co więcej, rozwiązanie jest na tyle elastyczne, że uwzględnia możliwość współpracy magistrali z komputerami wyposażonymi w różne rodzaje procesorów (Intel, AMD, Cyrix, PowerPC). Istotną cechą PCI jest także jej wysoka zgodność pomiędzy poszczególnymi wersjami, jak i rozwiązań pochodnych (np. PCI X) przejawiająca się tym, że urządzenia mogą pracować zarówno w starszych, jak i nowszych gniazdach – pod warunkiem, że są dopasowane napięciowo.

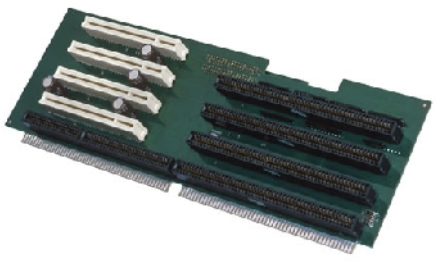
Zorro to nazwa magistrali rozszerzeń. Oferuje ona obsługę DMA (Direct Memory Access), bus mastering (karta rozszerzeń może sama inicjować transmisję, bez pośrednictwa procesora lub kontrolera magistrali, także bezpośrednio do innej karty) oraz autokonfigurację i automatyczne przypisywanie adresów (odpowiednik wprowadzonego w 1995 roku przez Microsoft mechanizmu Plug And Play). W sloty te wyposażone były tylko modele z linii high-end (A2000, A3000, A4000). Dla komputerów nieposiadających magistrali Zorro niezależni producenci stworzyli odpowiednie płyty rozszerzające je o te gniazda, podłączane do złącza karty turbo. Wyróżniamy trzy generacje slotów Zorro:

Zorro I – gniazdo rozszerzające Zorro wprowadzono w roku 1985 w komputerach Amiga 1000. W rzeczywistości nazwa ta dotyczyła pierwotnie płyty głównej, a nie magistrali, ale przyjęło się nazywać tak samą magistralę.

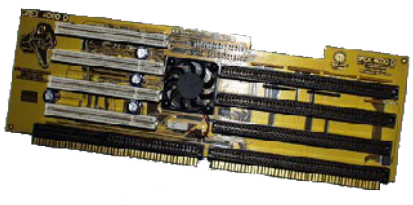
Zorro II – wprowadzono w roku 1987 w komputerach Amiga 2000. Zorro II była niejako buforowanym rozszerzeniem magistrali procesora MC68000, oferującym obsługę bus masteringu. Procesor mógł bezpośrednio adresować każde urządzenie jako pamięć RAM. 16-bitowa magistrala oferowała też protokół Autoconfig, dzięki któremu oprogramowanie automatycznie odnajdywało kartę, określało jej typ, po czym przydzielało adresy i przerwania.

Zorro III – magistrala zaprojektowana przez Dave'a Haynie i wprowadzona w roku 1990 w komputerach Amiga 3000, stosowana też w komputerach Amiga 4000. Zorro III, w odróżnieniu od poprzedniczki, miała 32-bitową szynę danych. Dla zgodności ze starszymi kartami użyto jednak tego samego, 100-pinowego gniazda, co wymusiło multipleksowanie szyn danych i adresowej, co z kolei komplikowało budowę kart i obniżało osiągi. Magistrala pracowała asynchronicznie względem procesora i oferowała tzw. arbitraż, co umożliwiało użycie kilku kart w trybie bus master jednocześnie.

Zorro IV – magistrala Zorro IV nie jest bezpośrednio związana ze standardem Zorro, lecz trzyma się jego wytycznych, oferując nawet coś ponadto. Jest to pomysł konstruktorów z firmy Elbox. Pozwala ona na podłączenie płyty rozszerzeń Mediator PCI ZIV, co w rezultacie daje możliwość korzystania jednocześnie ze slotów PCI i Zorro II.



G-Rex prototyp



G-Rex – wersja finalna

	Zorro	PCI	Uwagi dla Zorro	Uwagi dla PCI
Karty graficzne	TAK	TAK	• Najlepsza karta graficzna ma tylko 4 MB pamięci graficznej. • Możliwość uruchomienia Boot Menu Amigi przez kartę graficzną	• Dostępność do szybkich i tanich karty Voodoo, Radeon
Karty dźwiękowe	TAK	TAK	• Wiele kart oferuje sprzętową dekompresje MP3	
Kontrolery SCSI	TAK	TAK	• Możliwość uruchamiania systemu z dysku SCSI	• Brak możliwości uruchamiania systemu z dysku SCSI
Kontrolery USB	TAK	TAK	• Kontroler Deneb posiada własny BIOS, zapewnia możliwość pracy klawiatury i myszki bez sterowników.	• Wymagane są sterowniki do pracy USB, co w przypadku Spidera objawiało się problemami licencyjnymi ze stosem Poseidon
Kontroler SATA	NIE	TAK		
Kontrolery IDE	TAK	NIE		
Kontrolery multi I/O	TAK	TAK	• Współpraca z systemami AmigaOS, Linux, MorphOS, Unix	• Współpraca tylko z AmigaOS i MorphOS.
Dekodery MPEG	TAK	NIE	• Trudne do zdobycia i drogie. Dla odpowiednio szybkich Amig zbędne.	
Sprzętowe emulatory PC/Mac/Atari	TAK	NIE	• Trudne do zdobycia, drogie i dosyć wolne.	

Tabela przedstawia rodzaje urządzeń, jakich możemy użyć zarówno z mostkami PCI, jak i slotami Zorro.

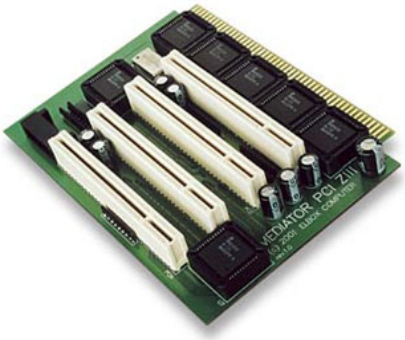
starą grę na rozbudowanej Amidze. Aby karta graficzna wyświetliła obraz w trybie PAL, trzeba uciekać się do sztuczek – obraz video Amigi sprzężamy z kartą TV wpiętą w Mediatora. Zorro za to zadowoli się aktywatorem slotu video i scandoublerem wpiętym w ten slot. Jest to jednak (znowu) dosyć droga zabawka. Jednak Zorro może być również „tanie". Załóżmy, że potrzebujemy jedynie szybkiego portu szeregowego oraz równoległego, do tego jakiś prosty kontroler IDE umożliwiający podłączenie dodatkowych dysków twardych (zakładam, że albo nie mamy Fast ATA, albo jej mieć nie chcemy, albo ją mamy i jest to za mało), prosty Multiface card + tandem IDE i możemy się cieszyć zaspokojeniem naszych wymagań.

Reasumując, Zorro II jest bardzo ciekawą magistralą o dużych możliwościach, pod pewnymi względami nawet większych niż PCI (możliwość uruchamiania z USB, slot video + scandoubler). Jest to jednak archaiczna, nawet jak na amigowe warunki, technologia, która w dodatku jest niesamowicie droga. Karta graficzna Picasso IV w tej chwili kosztuje ponad 1000 zł i jest trudna do zdobycia. Z drugiej strony jest Mediator, który nawet nowy, wraz z wszystkimi kartami PCI, tyle nie kosztuje. Oferuje on dużo większą szybkość niż Zorro i jest zdecydowanie nowocześniejszy. Do niego możemy wpiąć nawet kartę Radeon 9200, co już Amidze daje całkiem niezłego kopa. Samemu jednak trzeba się zastanowić, czego chcemy i w co zamierzamy inwestować. Rozbudowa Amigi 1200 o sloty rozszerzeń to niezbędna inwestycja, jeśli chce się używać komputera z wygodą i przyjemnością. Warto o tym pamiętać.

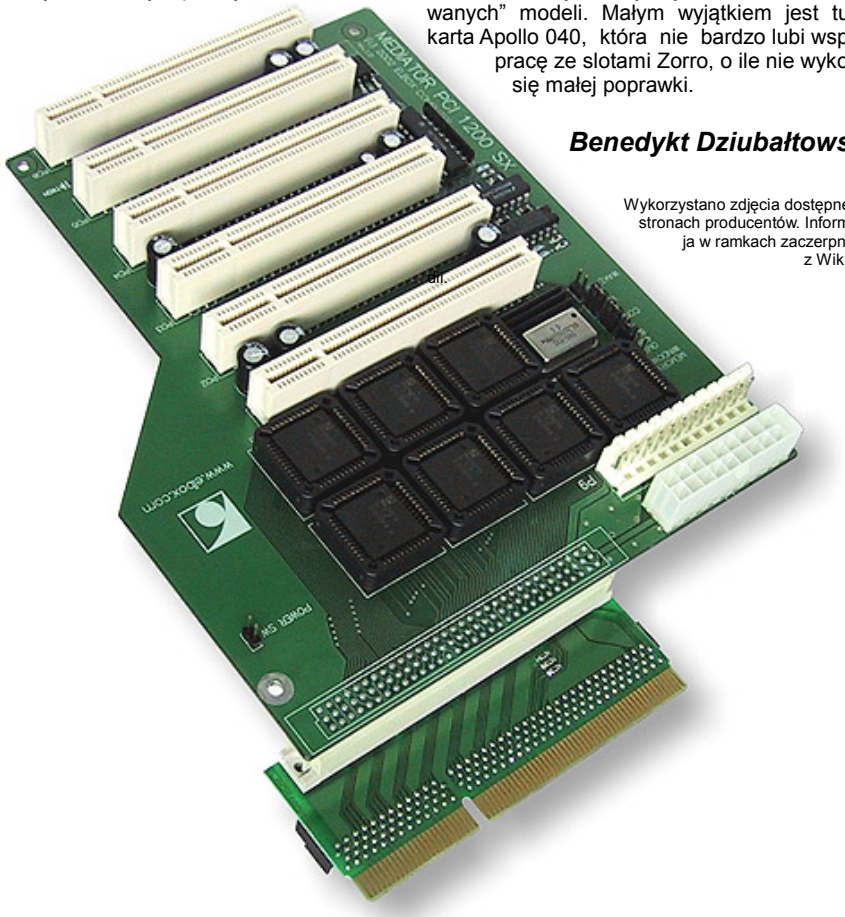
Na sam koniec dobra rada – jeżeli nasza Amiga posiada kartę turbo z procesorem 040 lub 060, to oczywiście warto wyposażyć ją w jakiekolwiek rozszerzenie – czy to Zorro, czy PCI – i chociażby podłączyć przez owe sloty zasilanie. Pozbywamy się problemów z prądem, które są zmorą użytkowników „owieżowanych" modeli. Małym wyjątkiem jest tutaj karta Apollo 040, która nie bardzo lubi współpracę ze slotami Zorro, o ile nie wykona się małej poprawki.

Benedykt Dziubałtowski

Wykorzystano zdjęcia dostępne na stronach producentów. Informacja w ramkach zaczerpnięta z Wikipedii.



Mediator PCI ZIII



Mediator PCI 1200 SX

Twardziel czy Compact Flash?

Pewnego pięknego dnia, usiadłem przy mojej Amidze 600 wyposażonej w DVD. Dłubiąc przy PC-Tasku uświadomiłem sobie, że choćbym dokonał nie wiadomo jakich cudów, to nie zainstaluje „łindołsa wisty" na dyskietce 720 kB. Trzeba było działać. Należało Amigę wyposażyć w jakiś dysk twardy lub jego odpowiednik. Stanąłem przed dylematem: dysk 3,5", 2,5" czy karta Compact Flash? Wybór nie jest prosty i z całą pewnością nie ja jeden się z nim mierzę. W tym artykule postaram się w miarę obiektywnie opisać wady i zalety poszczególnych rozwiązań. Tekst głównie skierowany jest do posiadaczy Amigi 600/1200. Pozostałe modele albo nie są wyposażone seryjnie w kontroler dysku twardego (Amiga 1000/500/500+), albo są to modele rzadziej spotykane i ich właściciele doskonale wiedzą, w co najlepiej wyposażyć swoje komputery (Amiga 2000). Należy również mieć na względzie fakt, że ceny podane w artykule obowiązywały w chwili pisania tego tekstu, a więc w 2011 roku.

Czy warto wyposażać Amigę w dysk twardy?

Zanim przejdziemy do porównań konkretnych rozwiązań, warto chwilę przystanąć i zastanowić się czy wydatki związane z rozbudową Amigi o dysk twardy są w naszym przypadku uzasadnione i konieczne. Z jednej strony większość z nas na tak postawione pytanie bez wahania odpowiada „tak", choć w gruncie rzeczy, wszystko zależy od tego, co chcemy robić i co robimy na naszej Amidze. Jeśli używamy Amigi „od święta", dla dwóch lub trzech gier, w które gramy średnio po kilkadziesiąt minut, to montaż dysku mija się z celem (może się to wydawać śmieszne, ale w obecnych czasach znaczna większość osób interesujących się Amigą ma niestety takie właśnie jej postrzeganie i właśnie w ten sposób ją użytkują). Jednak w każdym innym przypadku, montaż dysku twardego jest jak najbardziej wskazany.

Dysk twardy daje szereg zalet:

- Amiga po uruchomieniu ładuje system operacyjny. Nie trzeba korzystać z dyskietek systemowych. System działa znacznie szybciej i jest bardziej ergonomiczny i responsywny.
- Możemy dowolnie, w ramach możliwości sprzętu i pamięci, upiększać sobie wygląd Workbencha.
- Zdecydowaną większość gier możemy zainstalować na dysk twardy dzięki pakietowi WHDLoad.
- Granie w gry czy też używanie programów zainstalowanych na dysku twardym znacząco przyspiesza czas ich wczytywania, a tym samym zwiększa komfort używania Amigi.
- Odpada konieczność posiadania stada dyskietek. W dzisiejszych czasach, gdy dyskietka kosztuje około złotówki, koszt zakupu dysku twardego porównywalny jest z zakupem 30-70 dyskietek. Natomiast komfort użytkowania dysku twardego w stosunku do dyskietki jest nieporównywalny.

Aby zakup dysku twardego miał sens, musimy posiadać nieco rozszerzoną Amigę. Podstawa to 2 MB pamięci, choć przydałoby się nieco więcej, jeżeli myślimy o poważniejszym zastosowaniu oraz na przykład używaniu WHDLoad. Zalecany jest również Kickstart w wersji minimum 37.300 (dotyczy to Amigi 600). Da się również korzystać z dysku twardego na Kickstarcie w wersji 37.299, jednak wymaga to dyskietki rozruchowej, używanej tylko przy włączeniu Amigi do prądu.

Dysk 2,5"

Dyski 2,5" cechują się małymi wymiarami, w miarę bezgłośną pracą oraz dosyć wysoką ceną. Trzeba zaznaczyć, że ś.p. firma Commodore właśnie takie dyski dedykowała zarówno Amidze 600, jak i 1200. W czasach świetności tej firmy było to dosyć drogie rozwiązanie (choć wszystkie dyski twarde były wtedy drogie, to te były najdroższe). Mało tego, aby podłączyć taki dysk do Amigi 600/1200 należy zaopatrzyć się w specjalną taśmę 44-żyłową, która umożliwi podpięcie takiego dysku. Niestety owa taśma jest trudno dostępna. Jeżeli się uda nam ją znaleźć, to czeka nas koszt rzędu 20-30 zł. Za dysk twardy o pojemności 4 GB zapłacimy około 50-60 zł. Tak więc całość możne nasz kosztować nie więcej niż 100 zł wraz z wysyłką.

Dysk 2,5"

Co zyskujemy? Po pierwsze, możliwość instalacji dysku twardego w miejscu do tego przeznaczonym przez firmę Commodore. Dotyczy to oczywiście Amig w obudowie fabrycznej. Po drugie, przeważnie cichsza praca w porównaniu do dysku 3,5". Chyba najważniejszą zaletą jest prostota montażu takiego dysku. Rozkręcamy Amigę, wkręcamy sanki do dysku i spinamy taśmą dysk z komputerem. Z wad na pewno trzeba wymienić cenę. Dyski twarde, podobnie jak każda rzecz mechaniczna, zużywają się. W przypadku awarii takiego ustrojstwa jesteśmy 50 zł do tyłu, nie mówiąc o utracie danych. Poza tym nowego dysku twardego o pojemności 4 GB nie kupimy, a wkładanie do Amigi 600/1200 w podstawowej konfiguracji większego dysku, to tak zwany przerost formy nad treścią. Jeśli zaś mamy Amigę w wieży, to kupno dysku twardego 2,5" po prostu mija się z celem. Dysk twardy 2,5" proponuję montować w Amigach desktopowych, na których chcemy robić coś więcej niż tylko granie przy użyciu WHDLoad.

Dysk 3,5"

Jeden z najpopularniejszych rodzajów dysku twardego spotykany na rynku. Montaż nie jest tak bezproblemowy, jak u poprzednika. W Amidze 1200 trzeba wykonać kilka przeróbek, aby dysk zmieścił się w środku. Po pierwsze, żegnamy się z górną częścią ekranu – dysk musimy położyć na laminacie. Aby było to bezpieczne, dysk należny zaizolować. Kolejnym problemem jest zasilanie dysku. Trzeba wykonać to we własnym zakresie – albo za pomocą rozgałęziaczy i przelotek, albo za pomocą lutownicy doprowadzić napięcia +5V i + 12V do dysku. Najczęściej potrzebny prąd pobiera się z gniazda zasilania stacji dyskietek. Na sam koniec musimy zaopatrzyć się w przelotkę z dysku 2,5" na 3,5". Gniazdo dysku twardego w A600, jak i A1200, jest 44-piniowe, przystosowane do dysków 2,5". Dlatego właśnie wymagana jest stosowna przelotka.

Montaż dysku 3,5" w Amidze 600, to sprawa praktycznie beznadziejna. Możemy wyrzucić napęd dyskietek i w to miejsce zainstalować dysk twardy lub rozejrzeć się za dyskiem 3,5" typu slim i kombinować nad jego montażem w Amidze. Sytuacja wygląda zupełnie inaczej jeśli chodzi o Amigi w wieżach. Tutaj nie ma problemu. Jedyne co potrzebujemy, to wspomniana już przelotka z gniazda 2,5" na 3,5".

Jak każde urządzenie także i dysk twardy 3,5" ma swoje wady i zalety. Do zalet na pewno zaliczyć należy fakt, że dyski te są ogólnodostępne oraz tanie (ich cena jest nawet kilkakrotnie niższa niż dysku 2,5" o tej samej pojemności). Zrobiły furorę, jeśli chodzi o Amigi –

nawet te w obudowie desktop. Dobrze współpracują z każdą Amigą wyposażoną w kontroler dysku twardego. Wady to na pewno utrudniony montaż w „małych" Amigach desktop oraz nieco głośniejsza praca.

Karta Compact Flash

Karty Compact Flash (w skrócie nazywane CF) dosyć niedawno zagościły na amigowym rynku. Stało się to za sprawą tanich (około 30-40 zł) przelotek umożliwiających wpięcie karty CF do złącza dysku twardego Amigi. Taka karta ma bardzo wiele zalet. Jest mała i cienka, więc łatwo ją zamontować w Amidze 600/1200. Dodatkowo charakteryzuje się bardzo małym poborem prądu, a co za tym idzie nie obciąży zbytnio amigowego zasilacza – nawet tego bardzo leciwego. Jest również całkowicie bezgłośna – nie posiada silników ani innych ruchomych elementów. Amiga uruchomiona z karty CF nie wydaje żadnych dźwięków, no może poza klikaniem stacji dyskietek, ale to też można wyłączyć.

Niestety nie ma róży bez kolców. Jeśli często bawimy się Amigą i mamy dosyć rozbudowany sprzęt, to karta CF niestety odpada. Z pozoru jest to bardzo szybkie urządzenie (system wczytuje się błyskawicznie), lecz piękno rozwiązania znika, gdy kopiujemy bardzo dużo małych plików – karta CF robi to kilka razy wolniej niż dysk twardy. Nie pomaga tu zastosowanie kontrolera Fast ATA. Kolejną bolączką jest fakt, że każda karta CF ma ograniczoną ilość zapisów, po której przekroczeniu pojawiają się błędy na karcie i można ją wyrzucić do kosza. Amiga ma o tyle dobrze, że w przeciwieństwie do Linuksa czy Windowsa jej system operacyjny nie zapisuje ciągle danych, a w związku z tym karta nie zużywa się tak szybko.

Dysk 3,5"

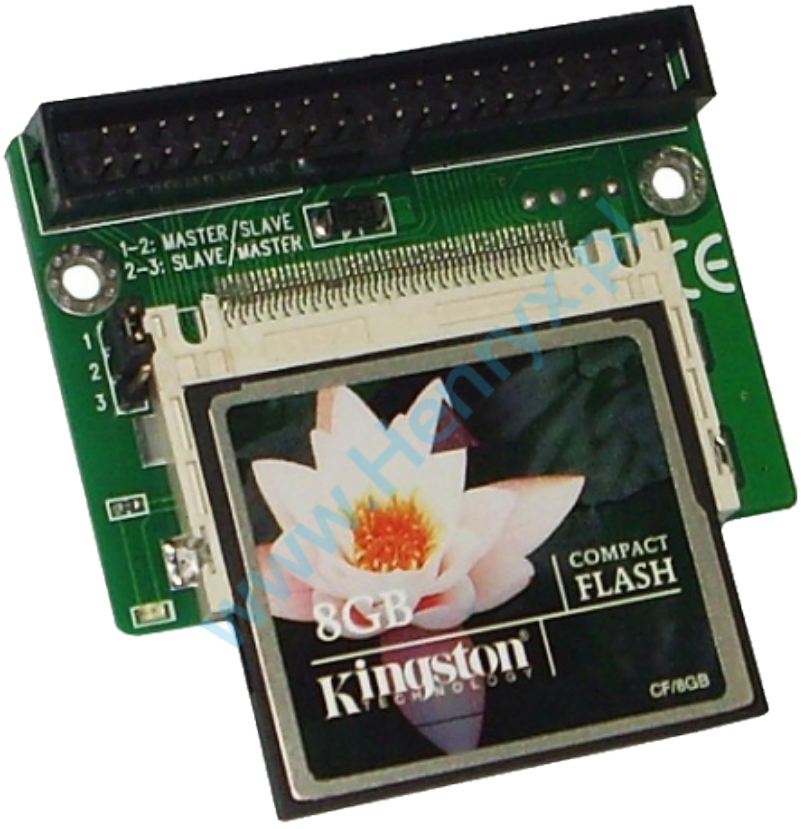
W kwestiach cenowych karta CF niestety przegrywa z pozostałymi rozwiązaniami. Aby podłączyć kartę CF do Amigi, musimy mieć specjalną przelotkę za wspomniane już 30-40 zł oraz samą kartę, której cena za pojemność około 4 GB wynosi 40-50 zł. Czasami też potrzebny jest dodatkowy kabelek łączący przelotkę z kontrolerem dysku twardego Amigi, co może podnieść koszt o kolejne 30-40 zł.

Karta CF w Amidze to idealne rozwiązanie, jeśli mamy komputer w obudowie desktop i interesuje nas przede wszystkim granie z wykorzystaniem WHDLoad. Warto też mieć na uwadze fakt, że nie każda karta CF będzie dobrze współpracować z Amigą.

„Tytułem endu.."

Jak widać, każde z trzech przedstawionych rozwiązań ma swoje wady i zalety. Mam nadzieję, że po przeczytaniu tego artykułu każdy będzie mógł łatwiej zdecydować się na któreś z opisanych rozwiązań. Ja jestem fanem dysków 2,5/3,5". Akurat poza WHDLoad Amigę używam do wielu innych celów, dlatego też karta CF nie zdaje u mnie egzaminu. Ale jest to oczywiście moje zdanie i każdy ma prawo się z nim nie zgodzić.

Benedykt Dziubałtowski



Adapter CF - IDE

Adapter Compact Flash II

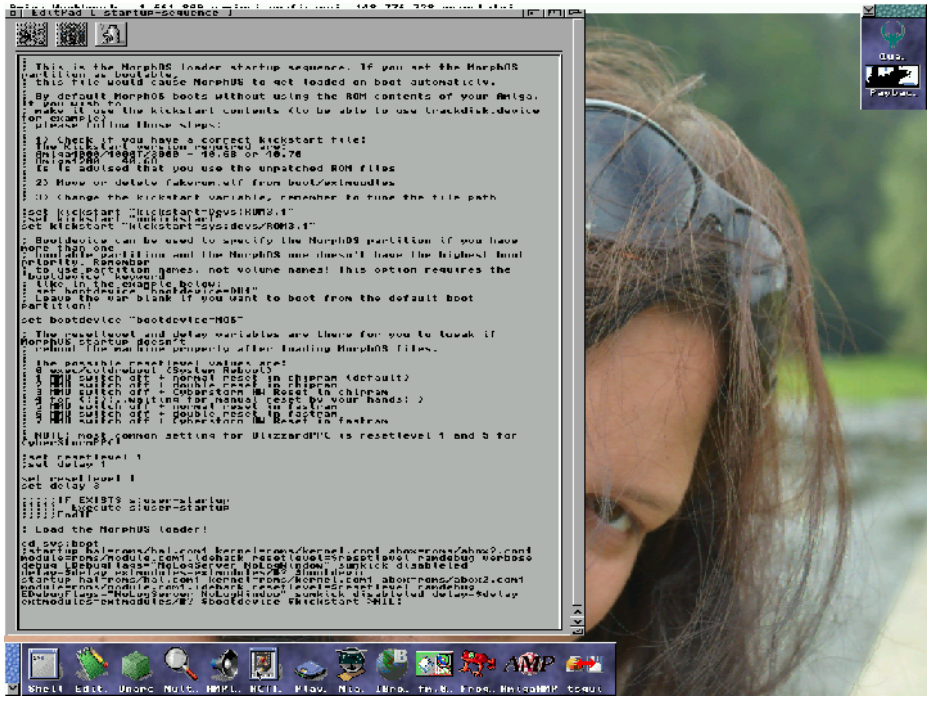
Instalacja systemu MorphOS 1.4.5. na A1200

MorphOS to pierwszy system operacyjny (nie licząc Linuksa) stworzony specjalnie dla Amig klasycznych wyposażonych w procesor PPC. Na pierwszej publicznej wersji 0.4 nie działało zbyt wiele oprogramowania, jak i niewiele było oprogramowania natywnego, które potrafiłoby w pełni wykorzystać możliwości systemu. Tak więc przez długi czas była to raczej ciekawostka niż nadająca się do użytku wersja systemu. Sytuację zmieniło dopiero pojawienie się wersji 1.4.5 (która jak dotąd jest ostatnią i nie zanosi się na pojawienie kolejnej). Do tego momentu baza oprogramowania, jak i doświadczenie, nabyte przez zespół deweloperski podczas wdrażania systemu w wersji dla Pegasosa i Pegasosa II, były już znacznie większe, a co za tym idzie wersja dla „klasyków" stała się czymś więcej niż zwykłą odskocznią od firmowo dodawanego do każdej Amigi systemu. Czego może spodziewać się użytkownik?

- system działa natywne na procesorze PPC, tak więc w stosunku do AmigaOS 3.x działającego nawet na najszybszej karcie 060, przyrost prędkości działania systemu (a nie tylko pewnych operacji wykonywanych przez niektóre aplikacje) jest wyraźnie zauważalny,
- pomimo tego, że system w wersji 1.4.5 nie jest już rozwijany, nadal pojawia się oprogramowanie dla nowszych wersji, które w znacznej mierze daje się na nim uruchomić,
- pewna część oprogramowania będzie działać lepiej na systemie MorphOS niż na AmigaOS 3.x – z prostej przyczyny – pewne błędy zostały wyeliminowane,

- dla MorphOS-a jest cały czas rozwijana dobra przeglądarka internetowa OWB, która pozwala odciąć się od leciwego, nierozwijanego IBrowse i normalnie przeglądać strony internetowe,
- system jest darmowy – wystarczy pobrać obraz płyty ze strony MorphOS-Team.net i zainstalować. W ten sposób przez dwie godziny możemy cieszyć się w pełni działającym systemem. Po upływie tego czasu system bardzo zwalnia i zachęca do zarejestrowania. Rejestracja jest również darmowa – to zwykła formalność, tak więc nawet powiedzenie „kupowanie kota w worku" się tutaj nie sprawdza.

Pozornie może się wydawać, że instalacja systemu MorphOS na Amidze klasycznej jest bardzo prosta. Wystarczy zrobić partycję, skopiować pliki i wykonać restart. Niestety w trakcie tych czynności może wydarzyć się kilka zdarzeń, które mogą wielu wprowadzić w zakłopotanie. Stąd też przyjrzymy się całemu procesowi krok po kroku.



Edycja sekwencji startowej systemu MorphOS

Wymagania

Zanim pobierzemy wersję instalacyjną systemu, warto przyjrzeć się jego wymaganiom. Może się okazać, że nasza Amiga po prostu nie spełnia pewnych wytycznych, co może być pierwszą przyczyną braku możliwości instalacji lub niedziałania systemu.

MorphOS w wersji dla Amigi klasycznej wymaga procesora PPC oraz karty graficznej (chipset AGA nie jest obsługiwany). W grę wchodzą więc A1200, A3000 i A4000 z kartami BlizzardPPC lub CyberStormPPC. W przypadku kart graficznych spektrum możliwości jest dosyć szerokie. MorphOS potrafi dogadać się z modelami opartymi na układzie Permedia2 i Permedia2V (a więc BVisionPPC i CyberVisionPPC). Do tego dochodzą karty wpięte w sloty Zorro (CyberVision 64/3D oraz Picasso IV) oraz PCI. Tym ostatnim trzeba poświęcić nieco więcej miejsca. Obsługiwane są modele Voodoo 3, 4, 5, a także karty oparte na układach SiS 6326 oraz SiS 305. Jedyną niedogodnością jest to, że karty PCI muszą być wpięte w mostek G-Rex. Niestety Prometeusz ani Mediator nie są obsługiwane i raczej nie należy się spodziewać, aby kiedykolwiek sytuacja uległa zmianie. MorphOS nie jest systemem pamięciożernym – do uruchomienia i prostej pracy wystarczy mu 32 MB pamięci. Nie sprawdzałem na mniejszej ilości, ale trzeba powiedzieć, że nawet jeżeli uruchomienie byłoby możliwe, to praca mija się trochę z celem. Moja konfiguracja to Amiga 1200 z Blizzardem PPC oraz kartą BVisionPPC i 160 MB pamięci.

Kopiowanie plików

Jeżeli nasza konfiguracja sprzętowa spełnia wymagania systemu MorphOS, możemy przystąpić do instalacji. Ze strony systemu pobieramy obraz ISO. Waży on dwadzieścia osiem megabajtów i skompresowany jest za pomocą LhA (w efekcie archiwum zajmuje niecałe 14 MB). Jeśli chcemy pobrać i wypalić obraz ISO na pececie, to warto do rozpakowania archiwum użyć programu *7Zip*. Użytkownicy Linuksa mogą użyć programu o swojsko brzmiącej nazwie *lha*.

Gdy już mamy płytę CD z systemem, należy przygotować dla niego partycję. Powinna ona

spełniać dwa kryteria:

1. Musi znajdować się w obszarze pierwszych czterech gigabajtów dysku twardego;
2. Powinna być sformatowana w systemie SFS.

SFS to skrót od *Smart File System*. Jest to system obsługi plików dużo lepszy oraz nowocześniejszy od FFS (*Fast File System*) stosowanego w AmigaOS 3.x. Archiwum z SFS znajdziemy na Aminecie. Mały problem mogą mieć użytkownicy systemów AmigaOS 3.0/3.1, gdyż trzeba ręcznie wpisać z klawiatury ID systemu. AmigaOS 3.5 oraz 3.9 ustawiają ID automatycznie. Gdy mamy już gotową partycję, kopiujemy pliki z płyty na dysk twardy. Niestety trochę to trwa – pomimo tego, że (jak wspominałem już wcześniej) na płycie znajduje się raptem kilkadziesiąt megabajtów danych.

Konfiguracja instalacji

Po zakończeniu procesu kopiowania nadchodzi czas na kolejny zestaw niezbędnych czynności. W katalogu *boot/extmodules* odnajdujemy i kasujemy plik **fakerom.elf**. Następnie bierzemy pod nóż *startup-sequence* i otwieramy go w dowolnym, nieco bardziej zaawansowanym edytorze tekstu (może być **Edit** z AmigaOS 3.9). Uwaga! Nie może to być **Ed** z AmigaOS 3.1 i starszych, gdyż źle łamie on zbyt długie linie, które niestety w *startup-sequence* MorphOS-a występują. Odnajdujemy linijkę:

```
set kickstart "nokickstart"
```

i zmieniamy ją na:

```
set kickstart "kickstart=Devs:ROM3.1"
```

O co tutaj chodzi? Zasadniczo MorphOS nie potrzebuje remapowania Kickstartu, ale dobrze jest to zrobić. Przyspiesza to działanie systemu. Bardzo ważne jest, aby użyć oryginalnego, niemodyfikowanego obrazu. Najlepiej zgrać ROM z naszych kości, które znajdują się w Amidze. Można to zrobić za pomocą jednego z wielu programów dostępnych na Aminecie. W pakiecie *Blizkick* również znajduje się taki program. Kickstart najlepiej przegrać do katalogu *Devs* i opisać jako **ROM3.1**.

Kolejną ważną linijką jest:

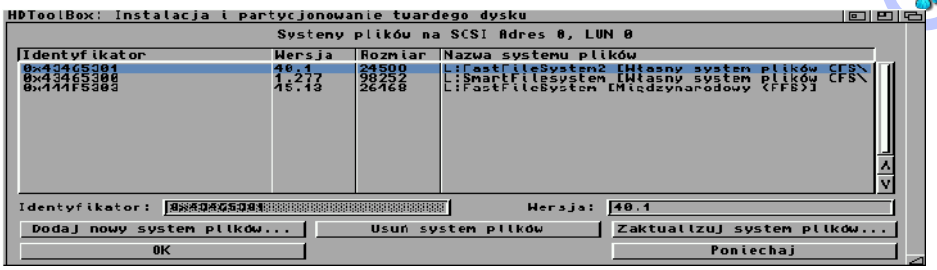
```
set bootdevice ""
```

Zmieniamy ją na:

```
set bootdevice "bootdevice=MOS"
```

Jeśli MorphOS nie jest wgrany na partycję o najwyższym priorytecie startu, to musimy prawidłowo poinformować system o tym, na której partycji się on znajduje. Bez tego system się nie uruchomi. U mnie oprócz MorphOS-a na dysku znajduje się AmigaOS 3.9 oraz AmigaOS 4.0. Partycję, na której siedzi „motylek" nazwałem po prostu MOS. Co jest bardzo ważne – wpisujemy fizyczną nazwę partycji, a nie jej nazwę będącą etykietą.

Zasadniczo możemy zapisać zmiany i wykonać restart. Jeśli MorphOS nie jest na partycji o najwyższym priorytecie startu (nie dokonaliśmy opisanej wcześniej zmiany parametru *set bootdevice*), to musimy wejść do Boot Menu i wybrać partycję z MorphOS-em. Po kliknięciu w przycisk *„Boot system"* rozpocznie się uruchamianie, po czym po chwili wykona się kolejny restart.



Trzy systemy plików na trzech różnych partycjach

Tutaj może pojawić się problem – MorphOS po pierwszym restarcie nie podniesie się. Przyczyn tego może być kilka. Po pierwsze, jeśli mamy Fast ATA i wpięte w secondary port jakieś urządzenie, to automatycznie możemy pożegnać się z MorphOS-em. Niestety trzeba wybierać: albo drugi port Fast ATA, albo MorphOS. Sprawa dotyczy również przelotek *4eide* i to zarówno tych produkcji firmy Elbox, jak i tych „made in garaż". Jeśli jednak nie mamy żadnego z podanych dobrodziejstw, a system nadal nie chce się uruchomić, to z powrotem wędrujemy do edycji *startup-sequence* i przyglądamy się ostatniej linijce (tutaj właśnie niezbędny jest prawidłowo odczytujący długie linie edytor tekstu).

```
startup hal=roms/hal.com1 kernel=roms/kernel.com1 abox=roms/abox2.com1
module=roms/module.com1 resetlevel=$resetlevel ramdebug verbose debug
EDebugFlags="NoLogServer NoLogWindow" sumkick disableled delay=$delay
extmodules=extmodules/#? $bootdevice $kickstart >NIL:
```

Ciąg **module.com1** zmieniamy na **module.com1.idehack**. Zapisujemy i wykonujemy restart. MorphOS powinien już odpalić.

Co dalej?

Niestety goły system jest niesamowicie ubogi. Dlatego w pierwszej kolejności należałoby go nieco stuningować. W systemie brak jest podstawowych narzędzi, takich jak stos TCP/IP, przeglądarka WWW. Poza tym warto jest zaktualizować biblioteki systemowe, MUI i doinstalować potrzebne nam programy.

Jak wspomniałem wcześniej, system działa tylko na procesorze PPC. W ogóle nie korzysta z procesora 68k. Jeśli uruchomimy jakiś program napisany dla AmigaOS i on się uruchomi, to będzie działał w trybie emulacji. W praktyce wygląda to tak, że każdy program 68k, na przykład *MiamiDx*, uruchamia się z prędkością światła i podobnie wykonuje wszelkie operacje związane z jego pracą. Mało tego, uzyskujemy dostęp do bardziej nowoczesnego i częściej uaktualnianego oprogramowania, takiego jak choćby wspomniane już wcześniej *OWB*, które bije na głowę wszystkie flagowe dla AmigaOS 3.x przeglądarki, czyli *IBrowse*, *AWeba* i *Voyagera* pod względem oferowanych możliwości. Tutaj jednak zderzamy się z rzeczywistością. Co by nie pisać o znaczących różnicach w prędkości między amigowymi PowerPC a procesorami linii 68k, to niestety aplikacje pisane pod MorphOS-a są pisane pod znacznie szybsze maszyny i z całą pewnością nie ma co oczekiwać ani spodziewać się, że ktokolwiek będzie je optymalizował pod leciwy sprzęt jakim jest klasyczna Amiga. *OWB*, pomimo tego, że pozwala normalnie przeglądać strony, działa na Amidze klasycznej wolno. Nie jest to jednak prędkość uniemożliwiająca pracę (jeśli mamy podkręcone PPC, to jak najbardziej da się tej przeglądarki używać), ale jeżeli ktoś oczekuje czegoś niesamowitego, wówczas nieco się rozczaruje.

Oczywiście nie samą przeglądarką człowiek żyje. Jest wiele programów, w tym emulatorów i gier, które albo nie powstały w wersji dla WarpUp, a działają pod MorphOS-em, albo mają swoje natywne wersje dla MorphOS-a, które z oczywistych względów działają dużo szybciej niż na 68k pod kontrolą AmigaOS 3.x. Przykładem takiej gry jest *„DukeNukem 3D"*. Wersja pod WarpUp dla AmigaOS 3.x nie obsługuje dźwięku – wersja dla MorphOS-a nie posiada tych ograniczeń. Podobnie przedstawia się sytuacja z emulatorem Macintosha *„Basilisk II"*, który działa natywnie. Na AmigaOS 3.x również działa natywne, ale tylko na procesorze 68k. Chyba nie muszę pisać, która wersja jest szybsza.

Jak widać korzyści i zalet jest sporo – i jeśli nawet jesteśmy fanami Linuksa czy AmigaOS to MorphOS-owi to nie przeszkadza. Grunt, że jest on zainstalowany na swojej partycji w systemie SFS, w obszarze pierwszych 4 GB. A to czy zaraz obok niego jest AmigaOS 3.x, 4.x czy Linux nie robi mu różnicy. MorphOS jest jedynie wrażliwy na Kickstart z AmigaOS 4. Trzeba pamiętać, że jeśli używamy tego systemu i chcemy włączyć MorphOS-a, to należy wyłączyć Amigę i odczekać aż zmapowany Kickstart AmigaOS 4 zniknie z pamięci.

Nie ma jednak róży bez kolców. Choć MorphOS nie gryzie się z AmigaOS 4.x, to nie lubi on, o czym już wspominałem, Fast ATA ani kontrolerów *4eide*. Dopóki nie używamy portu secondary, wszystko jest w porządku. Poza tym MorphOS słabo obsługuje amigowy sprzęt. Należy zapomnieć o Mediatorze, większości kart pracujących w slotach Zorro. Niektóre da się zmusić do działania, ale jest ich niewiele. Niemniej jednak, jeżeli Twoja amigowa konfiguracja opiera się na CyberVision w slotach Zorro lub BvisionPPC/CyberVisionPPC, to sądzę, że warto spróbować pobawić się MorphOS-em i porównać jego możliwości z AmigaOS 3.x czy 4.x. Jakby nie patrzeć, jest to część amigowej historii, z którą warto się zapoznać. Jeśli natomiast jesteś wrogiem MorphOS-a i uważasz go za konkurencyjny wyrób niszczący Amigę, to koniecznie musisz poznać ten system – trzeba poznać słabe strony wroga.

Tym optymistycznym akcentem kończę artykuł i pamiętajcie o ważnej zasadzie – bez względu czy na Amidze masz MorphOS-a, AmigaOS czy Linuksa, to Amiga zawsze RULEZ.

Benedykt Dziubałtowski

Odnośniki

- ***http://powerup.morphos-team.net/***

„King's Quest"

czyli o królach, smokach i tym podobnych

Witajcie wszyscy i siądźcie wygodnie bowiem historia, którą mam zamiar opowiedzieć, jest trochę długa. Jednak bez obaw – na tyle ciekawa, by rozerwać każdego amigowca: starszego i młodszego. Będę bowiem opowiadał o bajkach. Wiem, wiem – prawie nikt już dzisiaj nie wierzy w bajki. Świat stał się przewidywalny, przeliczalny i smutny jak stary, charczący wentylatorem pecet. Ale wspomnijcie na chwilę czasy Waszego dzieciństwa, kiedy ów świat był jeszcze pełen możliwości. Przypomnijcie sobie szum odpalanej z samego rana Amigi. Stańcie się na chwilę znów dziećmi, z napięciem wkładającymi dyskietki do swojego pierwszego komputera. Jesteście już tam? A więc, posłuchajcie historii Grahama, króla magicznej krainy Daventry, a także jego czcigodnej rodziny.

Dawno, dawno temu, za siedmioma górami, za siedmioma lasami... no dobra, po prostu daleko stąd, była sobie magiczna kraina Daventry. Jej władca, król Edward, rządził nią bardzo długo i sprawiedliwie. Do szczęścia brakowało mu tylko jednego – następcy, który mógłby odziedziczyć szczęśliwe królestwo. Wiedziony desperacją, Edward zawierzał kolejnym łajdakom oraz hochsztaplerom, którzy obiecywali mu potomka. W ten sposób pechowy władca postradał trzy magiczne przedmioty – lustro, tarczę oraz skrzynię, bez których Daventry skazane było na powolne wymieranie. Na misję odnalezienia zaginionych przedmiotów zostaje wysłany Graham, najdzielniejszy z rycerzy królestwa... Jak to w bajce, cała historia kończy się dobrze. Noszący na głowie charakterystyczną czapkę z piórkiem Graham pokona złą wiedźmę, olbrzyma oraz niegodziwego krasnoluda – po to, by w końcu odzyskać lustro, tarczę i skrzynię, stając się równocześnie następcą tronu królestwa Daventry.

Powstała w latach osiemdziesiątych gra oczarowała ówczesnych grafiką, doczekała się również wielu przeróbek i odświeżonych wersji. Chyba po raz pierwszy przeniesiono na ekran komputerów tak złożony i kolorowy świat rodem z baśni, zaś sposób sterowania oraz prezentacji bohatera stał się pierwowzorem większości gier przygodowych w nadchodzących dziesięcioleciach.

Gra w oryginalnej wersji mieści się na jednej dyskietce. Wersja odświeżona, wydana kilka lat później, ze znacznie poprawioną grafiką i interfejsem obsługi została upchnięta na cztery dyskietki.

King's Quest VI

Dobrze, tak więc kiedy ostatnio opuściliśmy naszego bohatera, był on już królem Daventry. Szkopuł w tym, że i on nie mógł doczekać się potomka, a to z prostej przyczyny – nie miał bowiem królowej. Tak więc głównym motywem drugiej części serii jest poszukiwanie takowej. Na szczęście z pomocą przychodzi biednemu Grahamowi magiczne lustro, które znamy z części pierwszej. Pewnego dnia nasz bohater widzi w nim piękną kobietę o imieniu Valanice, uwięzioną na najwyższym piętrze wieży z kości słoniowej. Rusza do nieznanej krainy o nazwie Kolyma, aby wyswobodzić damę.

Choć grafika *„King's Quest II: Romancing The Throne"* nie różni się prawie niczym od swojej poprzedniczki, przedstawiony w grze świat Kolyma jest jakby przyjaźniejszy, mniej dziki niż Daventry, zarazem też bardziej urozmaicony. Jak zwykle spotykamy postaci i miejsca znane nam głównie z bajek, legend oraz mitów. Fabuła obfituje również w drastyczne sceny. Oto, aby dotrzeć do pięknej Valanice, Graham będzie musiał, między innymi, uśmiercić spoczywającego w trumnie wampira. Gwarantuję, że scena owego mordu na złożonej z wielkich pikseli bladej postaci na długo pozostaje w pamięci.

King's Quest V

Fabuła trzeciej części serii początkowo zdaje się nie mieć związku z poprzedniczkami. Jej bohaterem okazuje się bowiem młody chłopak, służący w domu despotycznego i wysoce nieprzyjemnego w obejściu czarnoksiężnika Manannana. Dla Gwydiona, bo tak ma na imię ów młodzieniec, przez szereg lat cały świat ograniczał się do czterech ścian ponurego domostwa czarnoksięskiego chlebodawcy. Jednak pewnego dnia Manannan musi na kilka godzin wyjechać w interesach (czytaj: teleportować się, czynić świństwa gdzie tylko popadnie). Dla Gwydiona oraz kierującego nim gracza będzie to wyjątkowa okazja, by wydostać się spod uciążliwej kurateli.

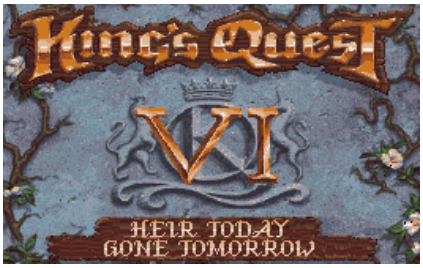
Piszący te słowa nie będzie nawet próbował ukryć, iż *„King's Quest III: To Heir Is Human"* jest jego zdaniem jedną z najlepszych i najoryginalniejszych gier w historii rozrywki komputerowej. I każdy, kto rozwiązywał zagadki, nerwowo zerkając co chwila na zegar odliczający czas do powrotu złego czarnoksiężnika, na pewno się ze mną zgodzi. A co dopiero powiedzieć o tym cudownym uczuciu wolności, które towarzyszy nam, gdy Gwydion wreszcie ucieka z niewoli wrednego dziada. Ups, napisałem Gwydion, a tymczasem chodzi o kogoś zupełnie innego... Młody chłopak jest bowiem tak naprawdę zaginionym dawno temu Alexandrem, synem Grahama i Valanice. Ale tego dowie się dopiero po pokonaniu morza, łańcucha gór oraz jednego wyjątkowo nieprzyjemnego smoka.

W sumie, dlaczego tylko faceci mają ratować świat? To pytanie najwyraźniej zadali sobie twórcy czwartej odsłony *„King's Quest"* o podtytule *„The Perils of Rosella"*, czyniąc jej bohaterką księżniczkę Rosellę, córkę króla Grahama oraz Valanice. Oto wspomniany władca Daventry doznaje zawału serca i pomóc mu może jedynie rzadka roślina lecznicza, która rośnie jedynie w odległej krainie Tamir. Zanim jednak Rosella zerwie magiczny owoc dla swego ojca, będzie musiała stawić czoła złej wróżce Lolotte.

Będę szczery – czwórka klasycznej serii trochę rozczarowała piszącego te słowa. Choć graficznie stanowiła ważny krok naprzód w stosunku do swoich poprzedniczek, jej fabule brakowało polotu. Również postać głównej protagonistki nie zapada w pamięć, choć jest to przecież jedna z pierwszych kobiecych bohaterek gier komputerowych w historii (Rosella tak naprawdę rozwinie skrzydła dopiero w *„King's Quest VII"*, ale to już nie nasza bajka). Jako ciekawostkę należy podać fakt, iż w trakcie poszukiwania magicznego owocu gracz odwiedzi pewną opuszczoną posiadłość. Roi się tam od duchów oraz różnych straszydeł, niektórych jakby żywcem wyrwanych ze znacznie późniejszej, słynnej interaktywnej gry pt. *„Phantasmagoria"*. Trudno się dziwić, skoro autorką obydwu gier jest ta sama osoba – Roberta Williams.

Wyobraźcie sobie następującą sytuację – wracacie pewnego razu do domu, powiedzmy, że po spacerze, patrzycie, a tu „bęc" – waszego domu nie ma. Taki właśnie widok zastał na początku części piątej serii król Graham. Po jego zamku w Daventry pozostała bowiem jedynie dziura w ziemi. Na szczęście zaraz pojawia się gadająca ludzkim głosem sowa o imieniu Cedric, która wyjaśnia osłupiałemu władcy, iż widok przed jego oczami wcale nie jest efektem całonocnej balangi, lecz skutkiem działań odzianego na czarno czarodzieja, który przy pomocy swojej magii skurczył zamek i zabrał go w sobie tylko znane miejsce. Cedric kieruje kroki króla do krainy o nazwie Serenia. Tam, z pomocą nieco już zdziecinniałego czarodzieja Crispina, obydwaj stają do walki z czarnoksiężnikiem Mordackiem.

„King's Quest V: Absence Makes The Heart Go Yonder" wniósł do serii przede wszystkim oprawę graficzną zupełnie nowej generacji. Gierka wyciskała z poczciwego ECS całkiem dużo. Świat Serenii w niej przedstawiony był bardzo zróżnicowany i miły dla oka, bohaterowie odwiedzali nie tylko miasta, pustynie, lasy, ale także archipelagi wysp, wreszcie też mroczne zamczysko głównego antagonisty. Z racji rozmiarów, porywanie się na tę grę bez twardego dysku wymagało od śmiałka nie lada odwagi. Rozczarowywał tylko wątek pierzastego towarzysza podróży króla Grahama. Cedric przydaje się głównemu bohaterowi stosunkowo rzadko, niewiele ma też ciekawego do powiedzenia.



A więc zamek wrócił na swoje miejsce, problemy rodziny królewskiej Daventry dalekie są jednak od zakończenia. Syn Grahama i Valanice, Alexander nie może się pozbyć z pamięci twarzy pięknej Cassimy, którą ujrzał podczas poprzedniej przygody. Pewnego dnia, natchniony wizją odbitą w magicznym lustrze, młody książę wyrusza w rejs, który kończy się dlań średnio szczęśliwie – sztormem, katastrofą i wyrzuceniem naszego bohatera na plaży nieznanej wyspy. „Kiszka", powiecie, a jednak nie do końca. Okazuje się bowiem, że nasz niedoszły Robinson trafił do królestwa Zielonych Wysp, księżniczka Cassima przebywa zaś całkiem niedaleko, uwięziona przez złego (a to niespodzianka!) czarnoksiężnika Abdula Alhazreda. Zajmie trochę czasu, nim książę zwiedzi wszystkie fantastyczne krainy, takie jak Labirynt Minotaura, czy Krainę Umarłych po to, by wreszcie odnaleźć swoją drogą ukochaną.

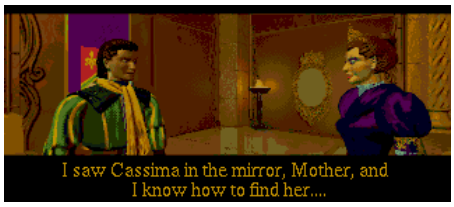
Szósta część serii *„King's Quest: Heir Today Gone Tomorrow"* to pod każdym względem absolutna perełka w swojej dziedzinie. Grafika, projekty postaci oraz lokacji, które zwiedza Alexander, są najwyższej próby, najbardziej jednak urzeka wartko płynąca fabuła, autorstwa wspomnianej już Roberty Williams oraz Jane Jensen, późniejszej twórczyni kultowej serii gier przygodowych pt. *„Gabriel Knight"* (ale to już nie nasza bajka).

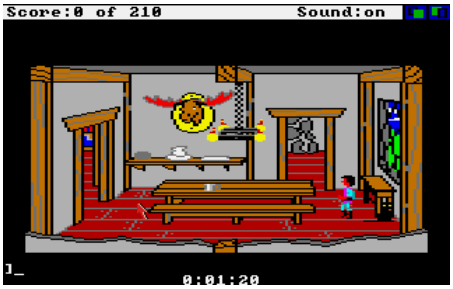
I na tym kończymy naszą przygodę z królem Grahamem i jego rodziną. Części siódma oraz ósma serii *„King's Quest"* nie posiadały już amigowych wersji. Na pocieszenie można jedynie dodać, iż owe dwie części nie były już tak dobre, jak ich poprzedniczki, głównie ze względu na duże zmiany w ekipie pierwotnych twórców.

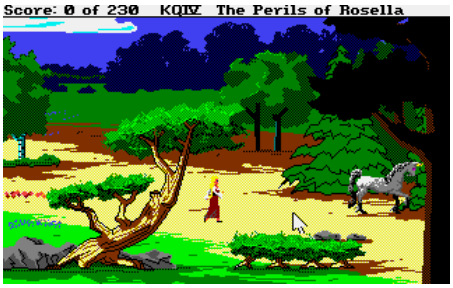
Piszący powyższe słowa, wiedziony słodkimi wspomnieniami dawnych czasów, celowo przemilczał mankamenty bądź złe strony serii *„King's Quest"*, stąd można wyciągnąć mylny wniosek, iż ich nie dostrzegł. Nie padło więc ani jedno słowo na temat wysokiego stopnia trudności gier oraz ślepych zaułków, w które gracz mógł zabrnąć, rozwiązując zagadki w nieodpowiedniej kolejności. Jakże łatwo można było zginąć, wykonując owe „Zadania Króla". Źle pokierowane postaci Grahama, Roselli bądź Alexandra mogły ponieść śmierć na tyle różnych sposobów, że owa ich śmiertelność stała się wśród współczesnych graczy wręcz przysłowiowa.

A jednak *„King's Quest"* pozostanie niekwestionowanym klasykiem wśród gier przygodowych swoich czasów, klasykiem wspomnień i cudownej przeszłości, klasykiem starej dobrej Amigi, wraz z którą odkrywaliśmy jako dzieciaki nowe światy.

Michał Leszczyński







Legion (SDL)

„Wiecie...” – zaczął Eioas, przeciągając się zamaszyście, aż drewniany stołek, na którym siedział, zatrzeszczał ostrzegawczo – „Wszyscy magowie twierdzą, że zwoje z czarami przepisywane są z pokolenia na pokolenie. Istnieje jednak prastare źródło tych czarów: Pierwsza Księga”. Siedzący obok troll Biue, wgryzający się w wielgachny – nawet jak na swoje nieskromne możliwości – udziec Gargoyla tylko milcząco skinął głową. Co prawda nie gadał z magami (a zwłaszcza z tym zakapturzonym pokurczem Ui), ale opowiadał mu kiedyś o tym wieśniak Oiopa. Gadał i gadał, aż Biue musiał go uciszyć swoją maczugą. Swoją drogą ten Eioas też już trochę za długo nawija... Troll namacał ostrożnie piszczel ogra leżący pod stołem, ale po chwili namysłu dyskretnie sięgnął po granitowy młot odwróconego plecami kobolda Alfe.

Paladyn Eioas prawdopodobnie nie zdawał sobie sprawy z tego, że głęboka myśl, którą podzielił się z resztą drużyny, wyświetli się na ekranach naszych komputerów, gdy tylko uruchomimy zrekonstruowaną przy pomocy C++ i SDL przez Studio Małych Form Komputerowych (Marek Lech), a przeportowaną na system MorphOS przez Pawła „stefkosa” Stefańskiego, niezapomnianą grę komputerową *„Legion”* – tę samą, którą pamiętamy sprzed lat. No, może niedokładnie tę samą, ale o tym za chwilę.

Wstęp, czyli... z czym do ludzi?

Przygotowany przez Stefkosa port to natywna MorphOS-owa wersja niezwykle ongiś popularnej gry strategiczno-RPG, którą stworzyło w 1996 roku GOBI software. Co prawda (zważywszy rozpowszechnienie tytułu) można by śladem Benedykta Chmielowskiego napisać „Legion jaki jest, każdy widzi”, ale niech tam – napiszę, o co w grze chodzi, choćby z myślą o naszych dzieciach i wnukach, które kiedyś będą z wypiekami na twarzy przeglądać siódmy egzemplarz papierowego PPA. Dowodzimy drużyną/drużynami dzielnych wojów (rekrutując ich spośród różnych ras lub specjalności), a naszym zadaniem jest... dowiedzieć się kto właściwie jest głównym mącicielem, a potem znaleźć go i mu dokopać. Oczywiście zadanie nie jest takie proste, a po drodze czeka nas wiele zabawy w postaci zdobywania i rozbudowywania osad, walki z drużynami trzech pozostałych śmiałków, wypełniania przeróżnych misji i rasowania i/lub ekwipowania swojej drużyny tak długo, aż w końcu staną się prawdziwymi twardzielami i zabijakami pierwszej wody.

Zanim po raz pierwszy uruchomimy grę (tu należy nadmienić, że dzięki uprzejmości autorów nie jesteśmy zmuszeni do przegrywania danych ze starego amigowego twardziela – wszystko znajduje się w archiwum) możemy się pobawić plikiem konfiguracyjnym o nazwie **Legion.-config**, znajdującym się w głównym katalogu programu. Ustawić można: tryb pełnoekranowy lub rozgrywkę w okienku, rozdzielczość, zastosowanie filtra grafiki, „bajery” graficzne w stylu podwyższonej jakości cieni, a także pogrubienie czy antyaliasing czcionek oraz prędkość samej rozgrywki.

Oprawa, czyli... z czym do ludzi!

Cóż... Napiszę tak: skoro wiemy, że port bazuje na oryginalnych grafikach, to należy się spodziewać, że cudów nie będzie. „I naprawdę, rzeczywiście tak jest”, jak mawiał jeden z moich ulubionych wykładowców. Grafika (poza kilkoma nie-tak-strasznie-okropnymi wstawkami renderowanymi) jest bardzo siermiężna, animacja potworna (ale o ile w przypadku napotykanych potworów można jeszcze uznać to za działanie celowe, to u naszych wojaków jest to wysoce naganne), a całość dobija jeszcze ogólna pikselowatość (no, chyba że ktoś odpala w okienku i 320x256). Muzyka – hmm... Moduły są takie sobie, ale da się słuchać. Gorzej z udźwiękowieniem. Sample są bardzo szczątkowe i słabe, a tło muzyczne lokacji (czyli kilka zapętlonych sampli odtwarzających się losowo) może przy dłuższym graniu prowadzić do nadmiernego wypadania zębów i/lub głuchoty (o czym, o ile dobrze pamiętam, ostrzegał dystrybutor na oryginalnym opakowaniu gry). ALE! Jeśli zbytnio skupimy się na oprawie, to umknie nam to, co najważniejsze, czyli mechanika samej rozgrywki.

Mechanika rozgrywki, czyli... z czym do ludzi?!

Grę prowadzimy zasadniczo w dwóch trybach. Pierwszy z nich to mapa przeznaczona do wydawania rozkazów poszczególnym legionom, na przykład obozowania, polowania czy ataku na osadę lub jednostkę przeciwników. Można również zajrzeć w ekwipunek naszych wojów, ustawić progi podatkowe dla posiadanych przez nas osad, rozbudować je o nowe budynki czy mury obronne, a także powołać w nich nowy legion. Drugi natomiast to tak zwana „Akcja w terenie”, podczas której wydajemy rozkazy poszczególnym wojownikom. Jeśli przebywamy gdzieś w „dziczy”, to możemy trochę pochodzić, ewentualnie pozbierać ziółka czy kamienie, natomiast jeśli znajdujemy się przy tym na terytorium jakiejś osady – możemy ją swobodnie zwiedzać (niezależnie czy należy do nas, czy nie), rozmawiać z napotkanymi wieśniakami czy wojownikami (czasem udzielą nam niezbędnych informacji), namówić ich do przyłączenia się do naszej drużyny, a nawet zabawić się w rzezimieszka i zabierać mieszkańcom ciężko zarobione pieniądze. Co bardziej niewyżyci mogą zaś sprawić im prawdziwą krwawą łaźnię (na początku gry jednak nie radzę tej zabawy – zbyt często będziecie widywać swoich wojowników obijanych niemiłosiernie przez zwykłego wieśniaka, w którego wstąpiły niespożyte siły witalne. Znacznie bezpieczniejszą metodą jest szycie do nich z łuku – gra nie uznaje ataku z dystansu za akt agresji, więc nawet naszpikowany strzałami mieszkaniec będzie spokojnie szedł przed siebie – oczywiście, dopóki zdrowia starczy). Wbrew pozorom te niehumanitarne akcje nic nas nie kosztują – mieszkańców nie ubywa, a przybywa pieniędzy, przedmiotów (czasem trafi się nawet jakiś rzadki i cenny) i doświadczenia.

Skoro już jesteśmy przy doświadczeniu, to jest ono niezwykle cenne. Po pierwsze – wojak z wysokim jego poziomem to prawdziwa maszyna do zabijania, a odpowiednio wyekwipowany staje się niemal niepowstrzymany. Po drugie –

punkty doświadczenia można zamieniać na pozostałe charakterystyki (dosłownie). Możemy za nie „kupować” podwyższenie poziomu punktów wytrzymałości, siły, odporności, szybkości czy magii (życie i magia muszą się oczywiście „doładować” do nowego poziomu – inaczej byłby to zbytek szczęścia). Zwolennicy farmakologii mogą też spróbować nafaszerować naszych wojów odpowiednimi miksturami czy zielskiem – przy większych dawkach efekty mogą jednak drastycznie odbiegać od założeń.

Atak na miasto wygląda dwojako. Jeśli pozbawione jest murów obronnych czy palisady, toczymy z obrońcami walkę wśród budynków. Jeśli zaś jest ufortyfikowane, to przyjdzie nam najpierw rozbijać mury (potrzebna w tym celu będzie albo katapulta i zapas kamieni, albo... (sic!) dobry łucznik. Tak tak, nawet najtwardsze granitowe ściany muszą ustąpić pod gradem strzał z łuku), a dopiero potem głowy chroniących je żołnierzy.

Co do samych questów – z reguły czeka nas najpierw uganianie się po przeróżnych osadach (mechanizm wygląda tak: W osadzie X słyszymy od kogoś o super mieczu, który znajduje się w jakiejś krypcie, ale nikt nie wie, gdzie to jest. Pytamy w innym mieście, ktoś kieruje nas – powiedzmy – na zachód. Szukamy na zachodzie, w końcu w jakiejś wiosce ktoś mówi, że powinni coś wiedzieć w mieście Z. Idziemy do Z, tam odsyłają nas do Y. Tam z kolei dowiadujemy się, że to w X mieszka mędrzec, który na pewno wie, o co chodzi. Udaje się dotrzeć z powrotem do X, gdzie – oczywiście nie za darmo – ktoś oznaczy nam kryptę na mapie), a potem zwiedzanie jakichś jaskiń czy katakumb. No, ewentualnie eksterminacja bandy hultajów terroryzujących okolicę. W późniejszym etapie gry tworzyłem specjalne „szybkie” legiony złożone z jednego wojownika, wyspecjalizowane w lokalizowaniu questów. Turlanie się całą hałastrą po kolejnych mieścinach nie jest ani łatwe, ani przyjemne, ani tym bardziej – tanie (żołnierze, którzy nie obozują w mieście, muszą coś jeść). Zwykle jednak warto odwiedzać odkryte lokacje – można nieźle się obłowić, a także znaleźć bardzo rzadkie przedmioty. Te z kolei mogą okazać się niezastąpione w zbliżającej się konfrontacji z Naczelnym Arcyłotrem – a ta prędzej czy później nastąpi, bo w końcu kto chce bez końca opędzać się od coraz śmielej poczynających sobie Wojowników Chaosu?

Bug buga bugiem pogania, czyli „z czym...” zresztą, wiecie sami.

Oryginalna gra nie była pozbawiona wad. Puryści zapewne ucieszą się słysząc, że większość udało się zachować, natomiast orędowników postępu uraduje fakt, że dodano kilka zupełnie nowych. Zacznę chronologicznie, od oryginalnych. Chyba największa z nich powiązana była z rozbudowywaniem postaci. W pewnym momencie (przy mocno napakowanych zawodnikach) każda kolejna modyfikacja parametrów kończy się katastrofą – ot, na przykład naszemu świetnie wytrenowanemu mordercy wręczamy super-miecz, a potem wyjmujemy mu go z ręki. Albo też, nie daj Boże, próbujemy go podleczyć liściem bobkowym albo wodą. Wujek dobra rada mówi: nie dotykać, chyba że czarami leczącymi. Inaczej z naszego super-żołnierza zrobi się superciapa. Kolejnym mocno irytującym elementem rozgrywki jest mocno losowe dobieranie scenerii. Heraklit powiedział podobno, że nie można dwa razy wejść do tej samej rzeki. Autorzy gry poszli dalej – nie można dwa razy wejść do tej samej osady. Budynki będą w tych samych miejscach, ale elementy scenerii (np. krzaczek, beczka czy zniechęcająco wyglądająca dziura w skorupie ziemskiej) będą już zupełnie gdzie indziej. Kłopot zaczyna się, gdy pagórek czy geosynklina złośliwie blokują wejście do sklepu, czy innej kluczowej budowli. Co prawda jutro też będzie dzień, ale kto da nam gwarancję, że i tym razem nic przed wejściem nie wyrośnie? Upierdliwe i bardzo frustrujące, szczególnie w połączeniu ze sposobem, w jaki przemieszczają się nasi bohaterowie (próbujący czasem z uporem godnym lepszej sprawy przedostać się przez ścianę, która stoi im na drodze). A skoro już przy przeszkodach terenowych jesteśmy, to wspomnę jeszcze o problemie, jaki mają nasi wojacy z przekazaniem sobie przedmiotu. Teoretycznie, jeśli stoją obaj na tym samym polu, to jeśli jeden upuści przedmiot, to drugi będzie go mógł podnieść z ziemi. W praktyce bywa z tym różnie (ciężko stwierdzić gdzie kończą się poszczególne „pola” w grze) i wygodniej jest nieraz sprzedać ów przedmiot w sklepie i kupić drugą postacią (oczywiście licząc się z marżą sklepikarza). Co prawda przedmiotami można się podzielić bez żadnego problemu w trybie mapy, ale najczęściej odczuwamy tę palącą potrzebę podczas akcji w terenie – szczególnie gdy mamy tylko jeden zwój z leczeniem, a nasz dyżurny czarodziej właśnie się „wyprztykał” z punktów magii.

Ale ja tu o starych bugach, które wszyscy dobrze znamy i kochamy, a miało być coś ekstra. Otóż, po pierwsze – jeśli podczas akcji w terenie ktokolwiek (nieważne „swój”, „obcy” czy „neutralny”) wpadnie do dziury/zapadni, gra zawiesza się, a nam pozostaje kontemplacja czarnego ekranu. Po drugie – klikanie na jakimkolwiek przycisku skutkuje przesunięciem się obszaru, po którym klikamy o kilka pikseli w górę, czyli jak sobie zdrowo poklikamy, to po chwili nie widać już nic. Odświeżenie ekranu (na przykład wyjście i wejście z powrotem do opcji dialogowych) naprawia problem, ale niesmak pozostaje. Na sam koniec pozostawiam ciekawostkę, którą trudno nazwać błędem, ale z pewnością jest to różnica względem oryginału. Mianowicie – w pierwowzorze można było (nie do końca uczciwie) nabijać sobie doświadczenie, gdy wszystko inne zawiodło – wystarczyło zwerbować wieśniaka do naszej drużyny, a następnie kazać mu iść gdziekolwiek i zaszlachtować go „w drodze”. W odświeżonej wersji *„Legionu”* ten numer nie przejdzie. Gdy tylko knypek zaczyna zbierać cięgi, natychmiast się odwraca i zaczyna uszczuplać nasze siły życiowe.

Znaczy się co... nie grać?

Nie do końca. Pomimo wszystkich wad, uproszczenia rozgrywki, ubóstwa graficznego i dźwiękowego, a także niewątpliwych (i momentami mocno frustrujących) błędów w samym kodzie gra ma jakiś nieodparty urok, dzięki któremu zawsze miło do niej powrócić i raz jeszcze uratować od niechybnej zagłady bezimienne, kwadratowe królestwo. Choćby jedyną nagrodą miał być obrazek skąpo odzianego młodziana w mieniących się złoto ochraniaczach, powiewającego nonszalancko zwieszonym przez ramię czerwonym sztandarem.

Konrad Czuba

- ładunek sentymentalny
- przyzwoita grywalność

- wołająca o pomstę do nieba oprawa
- irytujące błędy w kodzie

http://stefkos.amigazeux.net/Legion.lha

Nasza pierwsza gra

Kurs programowania z wykorzystaniem Visual Studio 2005 Express Edition i VBCC – część 2

Poprzedni odcinek zakończyliśmy przykładem pokazującym okienko naszej gry. Podobnie jak poprzednio, ustawiamy poprzeczkę wyżej i tym razem napiszemy namiastkę gry i ujrzymy ruch statku. Do tego brakuje nam jeszcze wielu rzeczy, więc do dzieła!

Najpierw zajmiemy się *timer.device*, za pomocą którego będziemy mogli bez problemu pokazać w naszym okienku ruch statku bez obawy, że przy innym odświeżaniu ekranu jego szybkość będzie inna od zamierzonej. Ponieważ programujemy pod systemem, to nie będziemy ingerowali bezpośrednio w hardware Amigi, lecz zapewnimy sobie do niego dostęp przez device'y. Nie jest łatwo zrozumieć, czym jest device – to nie jest biblioteka trzymająca pewne funkcje. Ja myślę o device jak o silniku, który sprawuje pieczę nad danym urządzeniem. Pozwala na skorzystanie z urządzenia przez próbę otwarcia danego device'a. Przez mechanizm zapytań (requestów) możliwa jest współpraca z urządzeniem, a możemy to robić w sposób synchroniczny. Tu do dyspozycji mamy funkcję *DoIO* z biblioteki *exec*. Funkcja ta czeka, aż nasze zapytanie się spełni i dopiero wtedy powraca, co jak łatwo się domyślić, może powodować problemy. Przykładem będzie długie czekanie, aż inny program zakończy swoje zapytanie z urządzeniem – bo jak wiemy, multitasking obowiązuje wszystkich i zagarnianie urządzeń może być bolesne przy współpracy z innymi programami. Mamy także możliwość wysyłania requestów asynchronicznie przez funkcję *SendIO* z *exec.library*. Ta funkcja powraca natychmiast, ale nie ma róży bez kolców – musimy poczekać na swoją kolej, sprawdzając czy nasz request został wykonany używając do tego celu sygnałów. Z punktu widzenia pisania gry z device'ów na pewno warto wymienić *audio.device,* odpowiedzialny za obsługę dźwięku czy też *gameport.device,* pozwalający dobrać się do joysticka od strony systemowej. No i oczywiście wspomniany wyżej *timer.device*, który zajmuje się mierzeniem czasu.

Zaczniemy od porządków. Nie będziemy pisali wszystkiego w jednym pliku, bo to powoduje większy chaos i każda zmiana w jednym pliku powoduje wydłużenie czasu kompilacji całego projektu. Przy małych projektach to może mieć małe znaczenie, ale myślę, że to poza dobrymi nawykami programowania także zwiększy czytelność kodu i poprawi hermetyzację danych. Przede wszystkim przeniesiemy funkcje (i niektóre zmienne) związane z oknem do oddzielnych plików: *window.c* i *window.h*. Utworzymy plik *types.h* zawierający nasze własne, często używane typy. Dodamy także dwa pliki *timer.h* i *timer.c*, które będą zawierać funkcje związane z *timer.device*. Stworzymy także pliki *input.c* i *input.h*, które będą związane z obsługą joysticka. Wszystkie te pliki umieścimy w tym samym miejscu, gdzie znajduje się *boxo.c*. Aby kompilator wiedział, że musi skompilować wszystkie te pliki, musimy zmienić odrobinę nasz plik *makefile*, w którym przechowujemy reguły kompilacji naszego projektu. Do zmiennych *OBJ* i *LINKOBJ* dodajmy *„window.o timer.o input.o”.*

```
OBJ = boxo.o window.o timer.o input.o
LINKOBJ = boxo.o window.o timer.o input.o
```

Po regule *box.o* napiszmy, jak kompilator ma stworzyć obiekty *window.o* i *timer.o*.

```
window.o: window.c
   $(CC) $(CFLAGS) window.c -o window.o
timer.o: timer.c
   $(CC) $(CFLAGS) timer.c -o timer.o
input.o: input.c
   $(CC) $(CFLAGS) input.c -o input.o
```

W pliku *types.h* zdefiniujemy nowy typ i będzie to wskaźnik na funkcję, która nie przyjmuje argumentów i nie zwraca wartości.

```
#ifndef BOXO_TYPES_H
#define BOXO_TYPES_H
//=======================================
#include <exec/types.h>
//---------------------------------------
typedef void(*PVF)(void);
//=======================================
#endif // BOXO_TYPES_H
```

Dla znawców C konstrukcja *#ifndef*, *#define* i *#endif* nie powinna sprawiać problemów. Jest to strażnik kompilacji, dzięki któremu kompilator nie zgłosi błędu powtórnej definicji, a konstrukcja *typedef* pozwoli nam zadeklarować wskaźnik na funkcję w uproszczony sposób, o czym przekonacie się, gdy będziemy usprawniać *boxo.c*.

Czas na *window.h* i *window.c*. W pliku nagłówkowym umieścimy funkcje i zmienne, które będą widoczne na zewnątrz.

```
#ifndef BOXO_WINDOW_H
#define BOXO_WINDOW_H
//=======================================
#include "types.h"
#define PENS_AMOUNT 2
//---------------------------------------
extern struct RastPort* g_pRpMain;
extern ULONG g_nWindowSignal;
extern ULONG g_tabPens[PENS_AMOUNT];
//---------------------------------------
extern BOOL signalsWindow(void);
extern int initWindow(void);
extern void killWindow(void);
//=======================================
#endif // BOXO_WINDOW_H
```

Funkcja *signalsWindow* to pod zmienioną nazwą nasza funkcja z poprzedniego odcinka, która zajmowała się odbieraniem sygnałów z naszego okna. Odrobinę ją rozszerzymy. Za rzeczy związane z otwarciem okna i inicjacją zmiennych będzie odpowiadać *initWindow*. Ostatnia funkcja będzie zamykać okno.

Spójrzmy na listing na stronie 17. Na samym początku warto zwrócić uwagę na fakt dołączenia pliku *window.h* zaraz w pierwszej linii – warto dołączać w ten sposób. Zapobiega to dziwnym błędom, które mogą dopaść początkujących, objawiające się kompilowaniem tylko niektórych modułów (przez moduł rozumiem tu pary plików typu *window.c* i *window.h*). Pisząc *static* przed *struct Window* uczyniłem zmienną *m_pWin* widoczną tylko w tym pliku, co uniemożliwi obcym modułom ingerencję w nasze okno. Do funkcji *signalsWindow* dołączyliśmy obsługę klawiatury, dodając stosowną flagę *IDCMP_RAWKEY* przy otwieraniu okna – dzięki temu możemy uzyskać informacje o kursorach, klawiszu A i ESC, co przyda nam się przy poruszaniu naszym statkiem. Tutaj także umieściłem pobieranie kolorów do naszej gry.

Zmierzmy się teraz z *timer.device*. Przedstawię plik nagłówkowy i w paru słowach go skomentuję.

```
#ifndef BOXO_TIMER_H
#define BOXO_TIMER_H
//=======================================
#include "types.h"
//---------------------------------------
extern ULONG g_nTimerSignal;
//---------------------------------------
extern int initTimer(void);
extern void killTimer(void);
extern void signalsTimer(void);
//=======================================
#endif // BOXO_TIMER_H
```

Jak widać powyżej, mamy tu kilka funkcji typu *init/kill*, które, jak łatwo się domyślić, będą odpowiadały za otwarcie/zamknięcie *timer.device* a *signalsTimer* zajmie się sygnałami pochodzącymi z naszego portu (w dalszej części zostanie wyjaśnione, skąd się wziął ten „nasz” port). Zewnętrzna zmienna *g_nTimeSignal* przechowuje numer sygnału. Kod pliku *timer.c* przedstawiony jest na stronie 18.

Skupmy się na funkcji *initTimer*. Aby zacząć pracę z urządzeniem, musimy najpierw spróbować je otworzyć. Do tego celu muszę mieć request, który przeważnie jest rozszerzeniem zwykłego *IORequest*. Nie inaczej jest w *timer.device*. Najpierw tworzymy port do komunikacji, a następnie robimy *IORequest* dla *timer.device*. Kolejnym krokiem jest otwarcie urządzenia. Jeśli wszystko się powiodło, to inicjujemy zmienne, które bedą używane, wypełniamy nasz *timerrequest* i wysyłamy go za pomocą funkcji *sendTimerReq*, która to ustawia, co ile mikrosekund ma przyjść powiadomienie. W tym przypadku jest to 1/50 sekundy (można by też napisać 1000000/5). Warto odnotować, że robimy asynchroniczne zapytanie do *timer.device*, bo nie chcemy blokować naszym programem systemu i dlatego użyliśmy *SendIO*. W funkcji *signalsTimer* odbieramy sygnały pochodzące z naszego portu i od razu wysyłamy ten sam request do urządzenia, aby uzyskać cykliczność 1/50 sekundy. Funkcja *killTimer* zajmuje się, jak nietrudno się domyślić, zamykaniem tego, co otworzyliśmy w *initTimer*. Bardziej doświadczonych programistów może zaskoczyć fakt, że ustawiamy zmienną *m_bTimerWasSent* na wartość *TRUE* po pierwszym wysłaniu requesta. Jest to potrzebne, gdyż może się zdarzyć, że użytkownik zaraz po ukazaniu się okna kliknie w gadżet zamykania i wtedy *AbortIO* może powiesić system. Jasne jest, że trzeba mieć niebywałe szczęście, aby to uczynić, ale wystarczy, że w naszym requeście będziemy czekali parę sekund, a to już wystarczy, by ktoś chciał opuścić nasz program zanim odbierzemy request. Po wywołaniu *AbortIO* wywołujemy *WaitIO*, aby system mógł dokończyć nasz request.

Większym zmianom ulegnie też plik *Boxo.c*, którego listing znajduje się na stronie 18. W kodzie przenieśliśmy funkcje i zmienne dotyczące okna. Zmieniliśmy init, dodając otwarcie

```
#include "window.h"
#include "input.h"
#include <proto/exec.h>
#include <proto/intuition.h>
#include <proto/graphics.h>
#include <intuition/intuition.h>
#define KEY_ESC 0x45
#define KEY_A 0x20
#define KEY_CURSOR_UP 0x4c
#define KEY_CURSOR_DOWN 0x4d
#define KEY_CURSOR_RIGHT 0x4e
#define KEY_CURSOR_LEFT 0x4f
ULONG g_tabPens[PENS_AMOUNT];
ULONG g_nWindowSignal;
struct RastPort* g_pRpMain = NULL;
static struct Window* m_pWin = NULL;
static BOOL m_bPensObtained = FALSE;
static void getPens(void);
static void freePens(void);
int initWindow(void)
{
     m_pWin = (struct Window*)OpenWindowTags(NULL,
            WA_Left, 0, WA_Top, 0,
            WA_Width, 320, WA_Height, 256,
            WA_CloseGadget, TRUE,
            WA_Title, (ULONG)"boxo",
            WA_Activate, TRUE,
            WA_DragBar, TRUE,
            WA_GimmeZeroZero, TRUE,
            WA_IDCMP, IDCMP_CLOSEWINDOW | IDCMP_RAWKEY,
            TAG_END);
            if (NULL == m_pWin)
            {
                        return -1;
            }
            getPens();
            g_nWindowSignal = 1L << m_pWin->UserPort->mp_SigBit;
            g_pRpMain = m_pWin->RPort;
            return 0;   //OK
}
void killWindow(void)
{
            freePens();
            if (m_pWin)
            {
                        CloseWindow(m_pWin);
            }
}
BOOL signalsWindow(void)
{
            BOOL bEnd = FALSE;
            while (TRUE)
            {
                        struct IntuiMessage* pMsg = (struct IntuiMessage*)
                        GetMsg(m_pWin->UserPort);
                        if (NULL == pMsg)
                        {
                                    break;
                        }
                        ULONG msg_class = pMsg->Class;
                        UWORD msg_code = pMsg->Code;
                        ReplyMsg((struct Message*)pMsg);
                        if (IDCMP_CLOSEWINDOW == msg_class)
                        {
                                    bEnd = TRUE;
                        }
                        else if (IDCMP_RAWKEY == msg_class)
                        {
                                    switch (msg_code)
                                    {
                                                case KEY_ESC: bEnd = TRUE; break;
                                                case KEY_A: g_bFire = TRUE; break;
                                                case KEY_CURSOR_UP: g_bUp = TRUE; break;
                                                case KEY_CURSOR_DOWN: g_bDown = TRUE; break;
                                                case KEY_CURSOR_RIGHT: g_bRight = TRUE; break;
                                                case KEY_CURSOR_LEFT: g_bLeft = TRUE; break;
                                                default: break;
                                    }
                        }
            }
            return bEnd;
}
static void getPens(void)
{
            typedef struct
            {
                        ULONG red;
                        ULONG green;
                        ULONG blue;
            } Color;
            Color colors[PENS_AMOUNT] =
            {
                        {0x00000000, 0x00000000, 0x00000000},          //BLACK
                        {0x7fffffff, 0x7fffffff, 0x7fffffff},          //GREY
            };
            int i;
            struct ColorMap* cm = m_pWin->WScreen->ViewPort.ColorMap;
            for (i = 0; i < PENS_AMOUNT; i++)
            {
                        ULONG red = colors[i].red;
                        ULONG green = colors[i].green;
                        ULONG blue = colors[i].blue;
                        ULONG pen = ObtainPen(cm, 0xffffffff, red, green, blue, PEN_EXCLUSIVE);
                        if( -1 == pen )
                        {
                                    pen = ObtainBestPenA(cm, red, green, blue, NULL);
                        }
                        g_tabPens[i] = pen;
            }
            m_bPensObtained = TRUE;
}
static void freePens(void)
{
            if (m_bPensObtained)
            {
                        int i;
                        for (i = 0; i < PENS_AMOUNT; i++)
                        {
                                    ReleasePen(m_pWin->WScreen->ViewPort.ColorMap, g_tabPens[i]);
                        }
            }
}
```

biblioteki *graphics* i dodając inicjację okna i *timer.device*. Funkcja *main* urosła o ustawienie początkowe zmiennych, które poza ustaleniem rzeczy związanych ze statkiem rysuje też ramkę ograniczającą nasz ruch. W pętli głównej oczekujemy sygnałów z okna, jak i z naszego requesta z *timer.device*. Sercem tej funkcji jest wywołanie funkcji za pomocą wskaźnika. My wywołujemy funkcję *putShip*, która odpowiada za ruch, rysowanie statku i sprawdzenie czy dotyka ściany. To dużo pracy, jak na taką funkcję i prawdę powiedziawszy powinniśmy ją rozdzielić na cztery: *undrawShip*, *moveShip*, *drawShip*, *collisionShip*. To pozostawiam jako proste ćwiczenie. Podpowiem, że *undrawShip* powinno usunąć statek z planszy, *moveShip* odpowiada za ruch statku, który jest możliwy tylko wtedy, gdy statek stoi nieruchomy. Zadaniem funkcji *collisionShip* ma być sprawdzenie czy dalszy ruch statku jest możliwy. Detekcja statku ze ścianą jest zrealizowana za pomocą sprawdzania koloru. Najpierw wyliczamy pozycję statku w nowym położeniu i odczytujemy w nim kolor. Jeśli natrafiliśmy na siwy kolor, to unieruchamiamy nasz statek, w przeciwnym razie dalszy ruch jest możliwy do wykonania i tylko wtedy zmieniamy pozycję naszego bohatera, jakim jest statek.

Tutaj drobna uwaga. Zamiast wywoływania funkcji *putShip* za pomocą wskaźnika na funkcję, byłoby łatwiej od razu umieścić wywołanie *putShip* i sterować przebiegiem programu za pomocą sprawdzania, co aktualnie powinno być wykonywane. Ale w ten sposób stracimy dwie rzeczy: elastyczność głównej pętli i niepowtarzalność kodu, która jest bardzo ważnym elementem w programowaniu umożliwiającym sprawne wprowadzanie zmian i zmniejszające czas konserwacji kodu. Poza tym dzięki temu, że wywołujemy kod za pomocą wskaźnika na funkcję, możemy w dowolny sposób zmieniać tenże wskaźnik i tym samym decydować, jaka funkcja ma być aktualnie wykonywana.

Przejdźmy teraz do modułu input. Będzie on odpowiedzialny za joystick. W jednym z kolejnych odcinków uzupełnimy go odpowiednimi funkcjami, a na potrzeby tego odcinka stworzymy niezbędne zmienne. Plik nagłówkowy:

```
#ifndef BOXO_INPUT_H
#define BOXO_INPUT_H
//=======================================
#include "types.h"
//---------------------------------------
extern BOOL g_bRight;
extern BOOL g_bLeft;
extern BOOL g_bUp;
extern BOOL g_bDown;
extern BOOL g_bFire;
//=======================================
#endif // BOXO_INPUT_H
```

i implementacja *input.c*:

```
#include "input.h"
//=======================================
BOOL g_bRight;
BOOL g_bLeft;
BOOL g_bUp;
BOOL g_bDown;
BOOL g_bFire;
```

Z pewnością warto będzie w tym module umieścić rzeczy związane z obsługą joysticka, więc na pewno trzeba będzie się zmierzyć z input.device bądź z gameport.device, ale to wszystko przed nami.

Jako ćwiczenie proponuję zmodyfikować nasz projekt i uczynić go jeszcze bardziej czytelnym. Na początek warto przyjrzeć się wywołaniu *SetAPen*. Mamy tam numer koloru, który otrzymujemy zaglądając do pierwszego elementu tablicy (*g_tabPens[0]*). O wiele jaśniej byłoby, gdybyśmy wiedzieli o jaki kolor nam chodzi bez uciekania się do zaglądania do funkcji, która pobiera pióra (*getPens*). Można to zrealizować na wiele sposobów, korzystając z *#define* bądź używając enumeracji – wybór zostawiam czytelnikowi. Przyjrzyjmy się funkcji *main* i skupmy się na kończącym go **return 0;**. Bardziej klarowne byłoby zwrócić jakąś stałą (na przykład **RETURN_OK**, która mówiłaby wprost, że kończymy program z sukcesem). Te kończące zero nie bez kozery jest na końcu funkcji *main*. Za pomocą zwracanej wartości można powiadomić OS o rezultacie naszego programu – co jest wykorzystywane w skryptach. Podobnie jak *main* należy zmienić funkcję *init*, modyfikując możliwe powroty z tejże funkcji. Pozwoli to wyłapywać potencjalne błędy w przyszłości. Dlaczego w przyszłości? Bo dla uproszczenia naszego projektu, nie informujemy użytkownika o ewentualnych błędach.

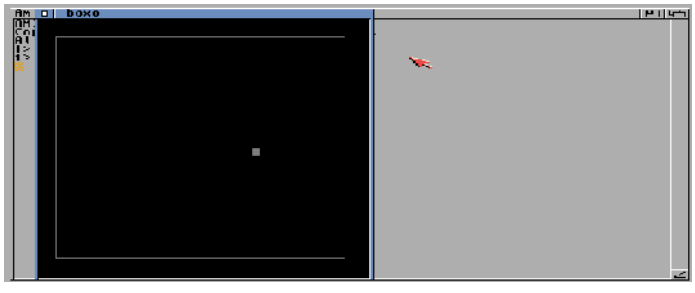
Kod pliku timer.c

```
#include "timer.h"
#include <proto/exec.h>
#include <devices/timer.h>
ULONG g_nTimerSignal;

static BOOL m_bTimerWasSent = FALSE;
static struct MsgPort* m_pMsgTimerPort = NULL;
static struct timerequest* m_pTimerIO = NULL;
static struct timeval m_tv;
static void sendTimerReq(void);
int initTimer(void)
{
            m_pMsgTimerPort = CreateMsgPort();
            if (0 == m_pMsgTimerPort)
            {
                        return -1;
            }
            m_pTimerIO = (struct timerequest*)
                        CreateIORequest(m_pMsgTimerPort, sizeof(struct timerequest));
            if (0 == m_pTimerIO)
            {
                        return -1;
            }
            LONG error = OpenDevice(TIMERNAME, UNIT_VBLANK,
                        (struct IORequest*)m_pTimerIO, 0);
            if (0 != error)
            {
                        return -1;
            }
            g_nTimerSignal = 1L << m_pMsgTimerPort->mp_SigBit;
            m_pTimerIO->tr_node.io_Command = TR_ADDREQUEST;
            m_pTimerIO->tr_time = m_tv;
            sendTimerReq();
            m_bTimerWasSent = TRUE;
            return 0;   //ok
}
void killTimer(void)
{
            if (m_pTimerIO)
            {
                        if (m_bTimerWasSent)
                        {
                                    AbortIO((struct IORequest*)m_pTimerIO);
                                    WaitIO((struct IORequest*)m_pTimerIO);
                        }
                        CloseDevice((struct IORequest*)m_pTimerIO);
                        DeleteIORequest(m_pTimerIO);
            }

            if (m_pMsgTimerPort)
            {
                        DeleteMsgPort((struct MsgPort*)m_pMsgTimerPort);
            }
}
void signalsTimer(void)
{
            while (TRUE)
            {
                        struct IntuiMessage* pMsg =
                                    (struct IntuiMessage*)GetMsg(m_pMsgTimerPort);
                        if (NULL == pMsg)
                        {
                                    break;
                        }
            }
            sendTimerReq();
}
static void sendTimerReq(void)
{
            m_tv.tv_secs = 0;
            m_tv.tv_micro = 20000;
            SendIO((struct IORequest*)m_pTimerIO);
}
```

Po zamianie magicznych liczb typu **0**, **-1** w funkcji *init*, można pokusić się o napisanie prostej funkcji, która na podstawie zadanego błędu zwraca informację przeznaczoną dla zwyczajnego użytkownika, bo chyba nikt nie lubi, gdy gra się nie uruchamia informując, że nastąpił błąd numer -1 bądź też nie informując wcale.

W kolejnym kroku zajmiemy się „kafelkami” (można spotkać się z nazwami: *tile*, *blok*, *klocek*). Są to bloki grafiki o ustalonych rozmiarach. Przykładowo kafelek 8x8 oznacza blok szeroki na 8 pikseli i wysoki na 8 pikseli. Dzięki nim oszczędzamy pamięć kosztem pewnej powtarzalności na ekranie gry. Wyobraźmy sobie ekran o rozdzielczości 320 pikseli szerokości i 256 pikselach wysokości. Daje nam to 1280 różnych kafli 8x8, więc jeśli użyjemy tylko 256 kafli, to sporo zaoszczędzimy. Oczywiście musimy gdzieś przechować informacje, który kafel gdzie ma się znajdować, czyli musimy posiadać mapę, która odwzorowuje planszę na ekranie. Dla naszego przykładowego ekranu taka mapa będzie miała 40 elementów szerokości i 32 elementów wysokości. Zagadnieniem map kafelkowych (*tile maps*) zajmiemy się w kolejnym odcinku.

Asman




```
#include "window.h"
#include "timer.h"
#include "input.h"
#include <proto/exec.h>
#include <proto/graphics.h>
#include <dos/dos.h>
struct IntuitionBase* IntuitionBase;
struct GfxBase* GfxBase;
PVF g_pFnc = NULL;
static int init(void);
static void loop(void);
static void close(void);
static void putShip(void);
static LONG m_nPosX = 0;
static int m_nVelocityX = 0;
static int m_nVelocityY = 0;
static LONG m_nPosY = 0;
static BOOL m_bShipStand;
int main(void)
{
            if (0 == init())
            {
                        m_nPosX = 200;
                        m_nPosY = 120;
                        m_nVelocityX = 0;
                        m_nVelocityY = 0;
                        g_pFnc = &putShip;
                        m_bShipStand = TRUE;
                        SetRast(g_pRpMain,g_tabPens[0]);
                        SetAPen(g_pRpMain,g_tabPens[0]);
                        SetBPen(g_pRpMain,g_tabPens[1]);
                        SetOutlinePen(g_pRpMain, g_tabPens[1]);
                        RectFill(g_pRpMain,16,16,288,224);
                        SetOutlinePen(g_pRpMain, g_tabPens[0]);
                        loop();
            }
            close();
            return 0;   //ok
}
static int init(void)
{
    IntuitionBase = (struct IntuitionBase*)OpenLibrary("intuition.library", 34);
            if (NULL == IntuitionBase)
            {
                        return -1;
            }
            GfxBase = (struct GfxBase*)OpenLibrary("graphics.library", 34);
            if (NULL == GfxBase)
            {
                        return -1;
            }
            if ( 0 != initWindow())
            {
                        return -1;
            }
            if ( 0 != initTimer())
            {
                        return -1;
            }
            return 0;   //ok
}
static void close(void)
{
            killTimer();
            killWindow();
            if (IntuitionBase)
            {
                        CloseLibrary((struct Library*)IntuitionBase);
            }
            if (GfxBase)
            {
                        CloseLibrary((struct Library*)GfxBase);
            }
}
static void loop(void)
{
     BOOL bEnd = FALSE;
     while (!bEnd)
     {
     ULONG signals = Wait(g_nWindowSignal | g_nTimerSignal | SIGBREAKF_CTRL_C);
                        if (signals & SIGBREAKF_CTRL_C)
                        {
                                    bEnd = TRUE;
                        }
                        if (signals & g_nWindowSignal)
                        {
                                    bEnd = signalsWindow();
                        }
                        if (signals & g_nTimerSignal)
                        {
                                    signalsTimer();
                                    (*g_pFnc)();
                        }
            }
}
static void putShip(void)
{
            SetAPen(g_pRpMain, g_tabPens[0]);
            RectFill(g_pRpMain, m_nPosX, m_nPosY, m_nPosX + 8, m_nPosY + 8);
            if (m_bShipStand)
            {
                        if (g_bLeft)
                        {
                                    m_bShipStand = FALSE;
                                    g_bLeft = FALSE;
                                    m_nVelocityX = -2;
                                    m_nVelocityY = 0;
                        }
                        else if (g_bRight)
                        {
                                    m_bShipStand = FALSE;
                                    g_bRight = FALSE;
                                    m_nVelocityX = 2;
                                    m_nVelocityY = 0;
                        }
                        if (g_bUp)
                        {
                                    m_bShipStand = FALSE;
                                    g_bUp = FALSE;
                                    m_nVelocityX = 0;
                                    m_nVelocityY = -2;
                        }
                        else if (g_bDown)
                        {
                                    m_bShipStand = FALSE;
                                    g_bDown = FALSE;
                                    m_nVelocityX = 0;
                                    m_nVelocityY = 2;
                        }
            }
            ULONG nX = 4+m_nPosX + 3*m_nVelocityX;
            ULONG nY = 4+m_nPosY + 3*m_nVelocityY;
            ULONG nColor = ReadPixel(g_pRpMain, nX, nY);
            if (g_tabPens[1] == nColor)
            {
                        m_nVelocityX = 0;
                        m_nVelocityY = 0;
                        m_bShipStand = TRUE;
            }
            else
            {
                        m_nPosX += m_nVelocityX;
                        m_nPosY += m_nVelocityY;
            }
            SetAPen(g_pRpMain, g_tabPens[1]);
            RectFill(g_pRpMain, m_nPosX, m_nPosY, m_nPosX + 8, m_nPosY + 8);
}
```

Free Heroes2

„Kłopoty zaczęły się w roku jednorożca, wraz ze śmiercią starego króla Enrothu, Lorda Ironfista. Władca pozostawił po sobie dwóch synów: Rolanda – dobrego, uprzejmego i honorowego oraz Archibalda – no cóż... Archibald był... Był... Eee... nie do końca tak dobry, jak Roland. Uprzejmość... Hmm, nie była jego mocną stroną, a o honorze powiedział kiedyś, że mogę go sobie... A zresztą mniejsza z tym, co powiedział. Zgodnie z wielowiekową tradycją wyboru nowego króla miał w takich warunkach dokonać Królewski Jasnowidz. Tak się jednak pechowo złożyło, że kilku kolejnych Widzących poniosło śmierć w dość tragicznych okolicznościach. W tej sytuacji Roland..."
– Dziadku, dosyć! Opowiadałeś nam już tę historię!
– Jak to, wnusiu? Kiedy?!
– Piętnaście lat temu, na moje trzecie urodziny!
– Cóż... W takim razie opowiem ją jeszcze raz. Tym razem za darmo i z opcjami specjalnymi!
– No dooobra...

Z okazji premiery szóstej już części cyklu o *„Bohaterach Mocy i Magii"* pomyślałem sobie, że przypomnę wszystkim, którzy jeszcze o tym nie wiedzą, że dzięki wysiłkom Andrieja Afletdinowa i jego ekipy oraz naszych dwóch zuchów, czyli Stefkosa i Widelca (a w przypadku wcześniejszych wersji portu – SixK) możemy natywnie zagrać na naszych ulubionych komputerach działających pod systemem MorphOS w część drugą, pochodzącą z roku 1996 (a właściwie – w przepisaną przy pomocy SDL, a korzystającą z oryginalnych danych wersję 0.6 silnika do tejże gry). Część z Was doskonale ją pewnie zna albo z wersji na PC, albo Mac (nierzadko okrutnie wymęczonej pod Shapeshifterem – nie każdy miał w końcu 060 i kartę graficzną). Tym, którzy w związku z tym chcą przerzucić stronę wspomnę tylko, że *„Free Heroes2"* to coś więcej (a także coś mniej) niż oryginał. Co mam na myśli? Hola, wędrowcze, nie tak prędko – zapraszam do lektury...

Sama gra stanowi połączenie strategii oraz gry RPG rozgrywającej sie w świecie fantasy. Gracz kieruje poczynaniami herosów (maksymalnie ośmiu), którzy to z kolei nie mogą ruszać się bez armii (choćby w takiej „armii" miał się znajdować jeden chłop). Zarówno herosów, jak i stworzenia walczące pod ich rozkazami możemy wynająć/kupić w zamku, ewentualnie mieście (które z reguły można do zamku rozbudować – zwiększa to jego możliwości obronne, jak i umożliwia dalszą rozbudowę). Czasem na mapie można natknąć się na budynek będący siedzibą danego typu jednostek, umożliwiający dodatkowe powiększenie naszej armii (za darmo lub tak, jak w zamku – kosztem złota). Skoro już przy tym jesteśmy – jeden bohater ma do obsadzenia pięć slotów na jednostki, w każdym z nich musi być minimum jeden żołdak danego typu. Co ciekawe, podziału jednej jednostki między kilka slotów (czasem się to przydaje) dokonujemy – w odróżnieniu od oryginału, w którym robiło się to z wciśniętym klawiszem Shift – przy pomocy prawego klawisza myszki.

W grze występuje sześć rodzajów zamków: Czarodziejki (Sorceress), Rycerza (Knight) i Czarodzieja (Wizard) – czyli sił dobra oraz Barbarzyńcy (Barbarian), Czarnoksiężnika (Warlock) i Nekromanty (Necromancer) – sił zła. Każdy z nich dysponuje szerokim wachlarzem jednostek o zróżnicowanych parametrach (co więcej, wiele spośród nich można dodatkowo ulepszyć, jeśli rozbudujemy budynek, w którym są rekrutowane). Mamy więc: stworzenia powolne, ale wytrzymałe (jak na przykład ogry), latające (gargulce), łuczników (elfy) czy posiadające szczególne umiejętności (jak na przykład smoki, wykazujące odporność na magię czy jednorożce, których atak potrafi oślepić przeciwnika). Zanim postawimy w zamku budynki, w których zakupimy najsilniejsze dostępne jednostki, musimy jednak trochę się napocić. Każdy z nich wymaga spełnienia określonych warunków – obecności w mieście innych struktur, a także dość sporej ilości surowców. A skoro już przy surowcach jesteśmy, to jest ich w grze sporo: Podstawowe to złoto (jakże by inaczej), drewno i ruda, ale daleko na nich nie zajedziemy (choć zamek rycerza jeszcze może sobie jakoś poradzi). Większość bu-

dynków będzie od nas wymagać dodatkowo kryształów, klejnotów, rtęci bądź siarki. Oczywiście surowce nie biorą się znikąd – trzeba je wydobyć z kopalni. Musimy więc odpowiednią znaleźć, a następnie przejąć (oflagować) i postarać się, żeby nie przejął jej zawsze pazerny na surowce konkurent. W sytuacjach awaryjnych zawsze można skorzystać z targowiska, ale przeliczniki za towary (nawet jeśli mamy kilka zamków i targowisko w każdym – „spready" są wówczas mniejsze) potrafią być zabójcze.

Wspominałem o walce? Nie? Być nie może. Prowadzimy ją (podobnie jak i całą rozgrywkę) w systemie turowym – raz my, raz przeciwnik. Pierwsze ruszają się jednostki najszybsze, a jeśli prędkości są równe, to zaczyna ten, kto zaatakował. Herosi nie biorą bezpośredniego udziału w walce, ale po pierwsze – ich współczynniki i zdolności mają wpływ na to, jak walczą żołnierze (dlatego warto na przykład inwestować bohaterowi w punkty ataku czy zdolności przywódcze, które zwiększają morale armii i dadzą nam szansę na dodatkowy ruch w turze), a po drugie – potrafią rzucać zaklęcia. No, może nie wszystkim najlepiej to wychodzi – mówię tu szczególnie o rycerzach czy barbarzyńcach, którzy znacznie lepiej czują się z mieczem w dłoni niż wymachując różdżką, ale odpowiedni czar rzucony we właściwym momencie potrafi, jeśli nie zdewastować siły nieprzyjaciela (a są takie zaklęcia), to przynajmniej znacząco przechylić szalę zwycięstwa na naszą korzyść. Oczywiście nic za darmo – jeśli wyczerpiemy punkty magii, to będzie je trzeba uzupełnić (same odnawiają się dość wolno). Najlepiej będzie uczynić to w przydrożnej... studni (tak, wiem że to trochę głupie) lub zatrzymując się na resztę dnia w mieście posiadającym gildię magii. Skoro już jesteśmy przy „wspomagaczach", to warto wspomnieć o artefaktach, które nasi herosi mogą nosić przy sobie. Jednym z podstawowych jest księga czarów (często znajduje się na „standardowym wyposażeniu" herosa. Jak się pewnie domyślacie, dla rycerzy i barbarzyńców jest to „wyposażenie opcjonalne" i wymaga dopłaty – a konkretnie zakupu „knigi" za pięć stów we wspomnianej gildii), w której zbierają się wszystkie zaklęcia, których się nauczymy – bądź to w kolejnych piętrach gildii, bądź w przydrożnych kapliczkach. Księgi – w odróżnieniu od innych artefaktów – nie da się przekazać innemu z naszych herosów. A przy okazji – nie ma znaczenia, że nasz bohater będzie jednocześnie używał np. miecza, tarczy, halabardy i morgensterna, może mieć nawet 14 par samych butów – każdy artefakt będzie aktywny. Uważajcie jednak – czasem można znaleźć artefakt, który działa zgoła odwrotnie niż byśmy sobie życzyli – przynosi pecha, zabiera nam pieniądze czy powstrzymuje neutralne jednostki rozproszone po mapie przed przyłączeniem się do nas. Pozbyć się takiego „kwiatka" wcale nie jest łatwo (ale zawsze można kupić sobie jakiegoś ostatniego łachmytę, podzielić się z nim wspomnianym artefaktem, a następnie podsunąć go przeciwnikowi, hłe hłe).

Ale my tu gadu gadu, omawiamy szczegóły rozgrywki, taktyki... a ani słowa jeszcze nie było o tak zwanej oprawie audiowizualnej. Pora więc nadrobić tę karygodną zaległość. Pierwowzór uruchamiał się na ekranie 640x480 pikseli, w 256 kolorach. W przypadku *„Free Heroes2"* widać w tej materii znaczący postęp – w opcjach możemy ustawić zarówno ekran, jak i okienko i właściwie dowolną rozdzielczość (nie polecam jednak wyższej niż 640x480 – gra potrafi wtedy czasami zdrowo się „zamyślić"). W porównaniu do części pierwszej widać odejście od stylistyki cukierkowo-bajkowej – grafika jest znacznie bardziej realistyczna (a już szczególnie – wygląd samych miast). W kolejnych częściach kontynuowano zresztą ten zdrowy trend. Elementy mapy, animacja jednostek – wszystko wygląda bardzo ładnie i cieszy oko. Efekty dźwiękowe są charakterystyczne i właściwie dają radę, choć można je było lepiej nagrać. Ciekawostką jest fakt, że o ile w oryginale na przykład stojąc przy kopalni będziemy słyszeć odgłosy prac pod ziemią (w postaci zapętlonego sampla), to w opisywanej wersji dźwięk odtwarzany jest tylko raz. Co do muzyki – to takowej (zapewne „jeszcze") w grze nie ma. Są za to puste opcje tam, gdzie w ustawieniach znajdowała się np. regulacja jej głośności. Moim zdaniem jednak muzyka nie jest szczególnie przydatna w trakcie rozgrywki (a na ekranie miasta może nawet irytować – chociaż, na przykład, w części trzeciej muzyka przy wejściu do miasta mi nie przeszkadza. Może to kwestia przyzwyczajenia albo może raczej jakości samej muzyki).

Skoro już podstawy mamy obcykane, to pora wziąć się za „mięso". Włączamy *„Free Heroes2"*, klikamy w *„New Game"* i... tu czeka nas niestety mega rozczarowanie – na dodatek podwójne: Po pierwsze – niedostępny jest tryb kampanii. Po prostu ta funkcja jeszcze nie została zaimplementowana (podobnie jak niektóre ustawienia w menu podczas gry czy kilka interesujących zaklęć). Co prawda tracimy niewiele – kampania jest tylko jedna, gdzie opowiadamy się po stronie Rolanda albo Archibalda i wprowadzamy go na tron bądź utrwalamy na nim uzurpatora. Dla zniesmaczonych tak kiepską różnorodnością jest jeszcze dodatek *„Price of Loyalty"* zawierający dodatkowe cztery kampanie (w tym dwie duże), ale póki co –

są one dla nas wciąż pieśnią przyszłości. Po drugie – gdy (z braku kampanii) zaczniemy się ratować pojedynczymi planszami, rychło zauważymy niewiarygodną wręcz tępotę komputerowego gracza. Nasi oponenci poruszają się po mapie cokolwiek chaotycznie, nie zawsze wykorzystują okazję, nawet jeśli spektakularnie się im podłożymy, a podczas walki w niczym nie przypominają „tego złośliwego sukinsyna, który w pierwszym ruchu zmasakrował nam połowę doborowych łuczników i skutecznie zablokował/oślepił resztę", którego pamiętamy i którego pokonanie naprawdę przynosiło satysfakcję. W efekcie średnio zaawansowany „Herołsiarz" przechodzi bez wysiłku praktycznie każdą mapę i to na poziomie *„Impossible"*. Nie o takie „Heroes" walczyliśmy, panowie... Szczęściem pozostaje jeszcze tryb gry wieloosobowej (niestety na razie tylko w trybie *„Hot seat"*, ale dobre i to), w którym możemy pokazać kolegom kto tu naprawdę rządzi.

Na koniec chciałbym w trzech zdaniach podsumować opisywaną grę: Po pierwsze – nareszcie nie musimy odpalać żadnych emulatorów żeby pograć na naszym komputerze w grę z serii *„HoM&M"*. Po drugie – możemy ją skonfigurować w stopniu daleko większym niż oryginał, „moddując" w stronę części trzeciej (np. kupowanie dostępnych jednostek za pośrednictwem jednego budynku – studni, możliwość oczekiwania w walce czy pozostawienia części części armii jako strażników kopalni), ale nie tylko. Po trzecie i ostatnie – gra niestety wymaga jeszcze sporo pracy i miejmy nadzieję, że autorom wystarczy determinacji do wydania wersji 1.0.

Mimo wszystko warto grę uruchomić (zwłaszcza, że dzięki serwisom typu *http://gog.com* zdobycie oryginału gry w wersji „rozszerzonej" to wydatek rzędu góra 10 dolarów) i na własne oczy przekonać się jak działa. Gdyby tylko komputerowa inteligencja w grze była trochę bardziej wymagająca, to bez wahania wystawiłbym grze solidną „czwórkę", a tak – tylko słaba trójczyna...

Konrad Czuba

- ciekawe połączenie gry strategicznej i RPG
- przyzwoita oprawa audiowizualna
- mnóstwo nowych opcji usprawniających rozgrywkę

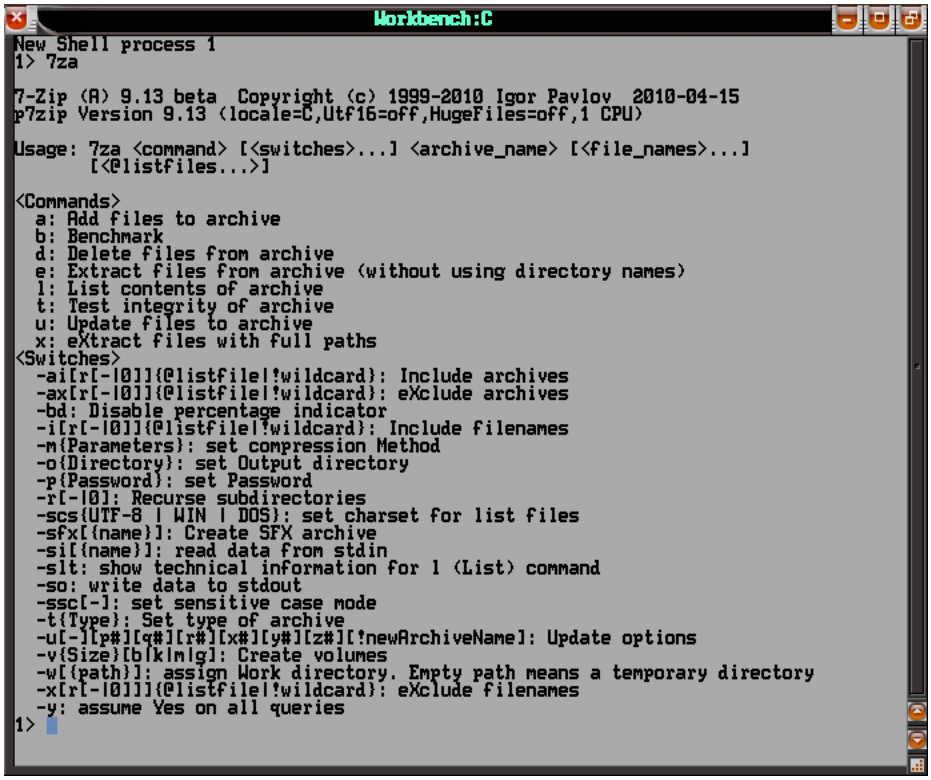
- liczne drobne niedociągnięcia, brak kampanii
- bardzo słabo grający komputerowy przeciwnik

„To GUI or not to GUI?”

czyli „rzeźbimy” skrypty w Directory Opus Magellanie

W 1998 roku firma GPSoftware z Australii wydała ostatnią amigową wersję swego sztandarowego programu *„Directory Opus Magellan II”*. Program ten niejako przełamał granice pomiędzy interfejsem okienkowym, do jakich przyzwyczaja zwykły desktop, a linią poleceń CLI. Nastąpiło to dzięki niezwykłej elastyczności Magellana w zakresie tworzenia nowych funkcji użytkownika, zapisywanych tak w postaci ikonek, jak i guzików menu belki tytułowej, guzików paska narzędzi, skrótów klawiaturowych czy funkcji menu kontekstowego, widocznego po wciśnięciu prawego przycisku myszy nad plikiem. I właśnie na przykładzie tego ostatniego, zwanego także menu kontekstowym, postaram się pokazać sposób tworzenia nowych funkcji w Magellanie. Przydatna będzie jako taka znajomość AmigaDOS i samego Magellana. Bez tego ani rusz.

Tytułem wstępu warto jeszcze wspomnieć o sposobie, w jaki Magellan rozróżnia pliki. Otóż Magellan potrafi rozróżnić pliki na nieograniczoną ilość sposobów (sic!). Może to być rozszerzenie w nazwie pliku, ale równie dobrze sam nagłówek pliku albo np. trzecia litera nazwy czy wręcz wielkość pliku. Co więcej, sposoby rozpoznania można ze sobą łączyć. Możliwości są w zasadzie nieograniczone. Jak nie pogubić się w tym wszystkim, może ktoś spyta? Powiem krótko: nie ma z tym żadnego problemu, a to dzięki trzymaniu się pewnych zasad. Na warsztat, jako przykład, wezmę po prostu folder. Magellan rozróżnia sam z siebie folder jako rodzaj pliku, więc sprawa definicji folderu jest prosta. Pytanie, jakie funkcje warto przypisać folderowi, a ściślej do menu kontekstowego folderu? Można różne, np. tworzenie obrazu ISO folderu, pakowanie w formacie ZIP, LZX, LhA lub 7z czy wręcz wypalanie folderu na CD, jak w przypadku folderu ze zdjęciami czy „empetrójkami”.



Tworzenie archiwum w formacie 7z

Za przykład niech posłuży mi modny ostatnio – i bardzo wydajny – archiwizer 7z. Jak do tej pory ukazało się kilka wersji tego programu dla AmigaOS, zarówno na Aminecie, jak i Sourceforge. Można wykorzystać dowolny z nich. Najnowsza dostępna wersja to 9.13 z Aminetu. Są to kompilacje ze źródeł linuksowych, pozbawione GUI, wyposażone w miarę bogaty zestaw argumentów dostępnych z poziomu CLI, czyli coś, co „młode tygryski lubią najbardziej”, bo przydaje się do „rzeźbienia skryptów” właśnie.

Popatrzmy na obrazek powyżej. Do archiwizacji katalogu będzie w zasadzie potrzebny nam tylko jeden z argumentów programu 7za, mianowicie argument:

a: Add files to archive *(Dodaj plik do archiwum)*

Ale że dobrze jest sprawdzić czy archiwum zostało utworzone bezbłędnie, więc przyda nam się także:

t: Test integrity of archive *(Sprawdź spójność archiwum)*

Zabieramy się więc za skrypt. Otwieramy edytor filetypów w Magellanie, wybieramy rodzaj pliku „Katalog” i w jego kontekstowym menu wklepujemy:

```
Funkcja:
        AmigaDOS    set nazwa {ou}
        AmigaDOS    C:7za a $nazwa.7z {f}
        AmigaDOS    C:7za t $nazwa.7z
        AmigaDOS    UnSetENV nazwa

Flagi: Output to window
```

Teraz krok po kroku „co i po co”:

AmigaDOS set nazwa {ou} – ustawia zmienną „nazwa”, zawierającą nazwę naszego folderu, argument Magellana **{ou}** zaś pozwoli mu na dalsze operacje na zaznaczonym folderze.

AmigaDOS C:7za a $nazwa.7z {f} – archiwizer 7za wkracza do akcji i pakuje archiwum (argument **a**) nadając jednocześnie plikowi wynikowemu nazwę oryginalnego katalogu i **.7z** jako rozszerzenie (przydaje się pod Windows).

AmigaDOS C:7za t $nazwa.7z – ponownie archiwizer 7za w akcji, tym razem sprawdzający integralność spakowanego archiwum (argument **t**).

AmigaDOS UnSetENV nazwa – usuwamy zmienną „nazwa” z pamięci.

Skrypt w zasadzie gotowy. Może jeszcze warto dodać docelowe miejsce, gdzie Magellan umieściłby nowo utworzone archiwum. Zakładam, że takie archiwum ma tymczasowy charakter, więc proponuję do tego celu katalog RAM:. Dopiszę więc jako pierwszą linijkę skryptu:

```
AmigaDOS cd RAM:
(tu reszta skryptu)
```

po czym jako ostatnią linijkę dodam:

```
Command ScanDir RAM: NEW
```

Polecenie **ScanDir** jest poleceniem wewnętrznym Magellana i otwiera nam nowy listerek z zadaną ścieżką – w naszym przypadku będzie to oczywiście lister z zawartością RAM:.

Podsumowując, efektem działania skryptu będzie:

- spakowanie dowolnego katalogu do formatu 7z,
- przetestowanie spójności tego archiwum,
- otwarcie listera zawierającego utworzone archiwum w formacie 7z.

Tworzenie obrazu ISO katalogu

Inny przykład funkcji menu kontekstowego katalogu to na przykład tworzenie jego obrazu ISO. Sprawa też niezbyt skomplikowana. Potrzebna będzie „pchełka” odwalająca niewdzięczną robotę, dostępna oczywiście na Aminecie.

Całość skryptu to niewiele więcej niż jedna linijka. Nie wiem czy w ogóle można mówić w takim razie o skrypcie w takim przypadku, ale ta jednolinijkowa funkcja tworząca obraz ISO katalogu działa jak trzeba. „Jednolinijkowy skrypt” wygląda więc następująco:

```
Funkcja:
   AmigaDOS mkisofs -o {ou}.iso -sysid AmigaOS -V {ou} -U {f}

Flagi: Output to window
       Window close button
```

I ponownie krok po kroku „co i po co”, choć tym razem opis samych użytych argumentów programu **mkisofs**:

-o {ou}.iso – ten argument ustawia wynikową nazwę obrazu na taką, jak nazwa katalogu, dodając jednocześnie rozszerzenie **.iso**

-sysid AmigaOS – zapisuje w obrazie ISO AmigaOS jako system roboczy

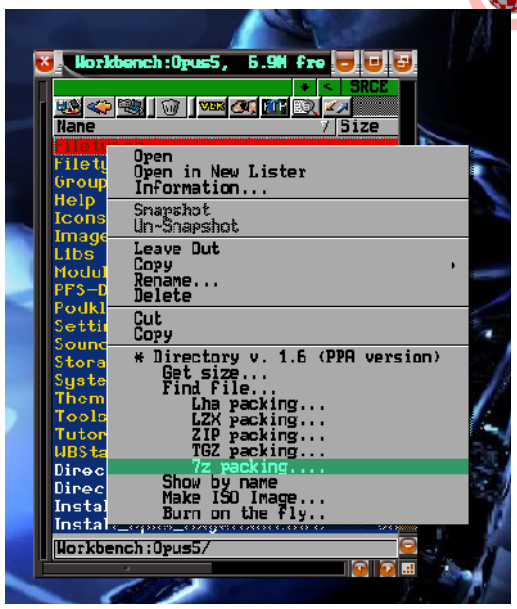
-V {ou} – zapisuje w obrazie ISO jego oryginalną nazwę, tj. nazwę katalogu

-U {f} – kombinowany zestaw kilku argumentów w jednym (odpowiednik **-l**, **-d**, **-L**, **-N**, **-relaxed-filenames**, **-allowlowercase**), szczegóły w instrukcji do programu **mkisofs**.

Program **mkisofs** zawiera dziesiątki argumentów CLI, opisywać je wszystkie mija się raczej z celem, wykorzystałem tylko kilka.

Archiwizacja katalogu w formacie LhA

Myślę, że format LhA jest na tyle popularny wśród braci amigowej, że nie wymaga specjalnego komentarza. Poniżej zamieszczam mój nieco bardziej rozbudowany skrypt do archiwizacji w tym formacie, podpięty oczywiście do rodzaju pliku definiującego katalog:




```
Function:
 AmigaDOS   cd RAM:
 AmigaDOS   set nazwa {ou}
 AmigaDOS   MegaEcho
 AmigaDOS   MegaEcho Directory filetype v 1.6 (PPA version) @ D. Gac) in action now...
 AmigaDOS   RequestChoice >ENV:wybor "Archiving with LHA" "Select mode:" "Aminet style|Best"
 AmigaDOS   IF VAL $wybor EQ 0
 AmigaDOS   SetEnv metoda -3
 AmigaDOS   SetEnv tryb -lh6-
 AmigaDOS   ELSE
 AmigaDOS   SetEnv metoda -2
 AmigaDOS   SetEnv tryb -lh5-
 AmigaDOS   ENDIF
 AmigaDOS   MegaEcho
 AmigaDOS   MegaEcho LHA packing with $tryb mode in progress...
 AmigaDOS   MegaEcho
 AmigaDOS   MegaEcho Creating archive: $nazwa.lha ... please wait !
 AmigaDOS   MegaEcho
 AmigaDOS   Lha -r -F $metoda -D1 a $nazwa {f}
 AmigaDOS   MegaEcho
 AmigaDOS   MegaEcho Completed... checking $nazwa.lha archive now...
 AmigaDOS   MegaEcho
 AmigaDOS   Lha -F t $nazwa.lha
 AmigaDOS   UnSetENV nazwa
 AmigaDOS   UnSetENV wybor
 AmigaDOS   UnSetENV metoda
 AmigaDOS   UnSetEnv tryb
 Command    ScanDir RAM:

Flags:
 Output to window
 Run asynchronously
```

Polecenie AmigaDOS MegaEcho to takie bardziej rozbudowane polecenie Echo. Zawiera w sobie kilka „bajerów”, które wykorzystuję w skryptach.

I tutaj pytanie edukacyjne „za 2 złote” do początkujących zwłaszcza: co takiego konkretnie robi powyższy skrypt?

Na koniec, myślę, że ciekawą funkcją jest wypalanie katalogu za pomocą programu **dvdrecord**. Kto pierwszy podejmie się zadania i „wyrzeźbi” skrypt?

I to by było na tyle. Polecam samodzielną zabawę w dodawanie rożnych funkcji (dla różnego rodzaju filetypów) i tym samym wzbogacanie desktopu pod kątem swoich własnych potrzeb.

A tytułowe pytanie *„to GUI or not to GUI?”* pozostawiam otwarte.

Dariusz Gac

Odnośniki

- ***http://amiga.sourceforge.net*** – Amiga Sourceforge – strona z oprogramowaniem open source.
- ***http://aminet.net/util/arc/p7zip-9.13-m68k.lha*** – archiwum z najnowszą wersją archiwizera 7zip przeznaczonym dla AmigaOS
- ***http://aminet.net/dev/gg/mkisofs-2.0bin.lha*** – program mkisofs służący między innymi do tworzenia obrazów ISO
- ***http://aminet.net/*** – główne repozytorium Aminetu

CD32

W całej historii konsolowego światka chyba nie ma przypadku, kiedy to startująca maszyna od samego początku swojego istnienia, musiała borykać się z tak wielkimi problemami na rynku, jak ostatnie w kolejności dziecko firmy Commodore – wizjonerska konstrukcja oparta o 32-bitową architekturę i napęd CD-ROM. Spadkobierca rozwiązań popularnych w latach 80-tych komputerów domowych spod znaku wielkiego C=, w obliczu uciekających deweloperów, niczym szczury w popłochu opuszczające tonący okręt, poniosła sromotną porażkę, poddając się azjatyckim konstrukcjom spod znaku Sony, Segi i Nintendo. Jako zwykły użytkownik, mimo tych wszystkich przeciwności, w krótkim czasie po tym jak nabyłem swój upragniony egzemplarz, pokochałem tę platformę na tyle mocno, aby w przyszłości patrzeć na wszystkie „nowomodne" rozwiązania konkurencji przez pryzmat użytkowania właśnie tego sprzętu.

Jak to wygląda?

Konsola wykonaniem nie zachwyca, ale też nie odpycha – szare, kanciaste pudełko z wywietrznikiem po prawej stronie obudowy i wypukłym napędem po lewej. Wzornictwo przywodzi na myśl konsolę Sega Megadrive typ I połączoną z tradycyjnym, kanciastym, desktopowym wyglądem Amigi 500 lub 1200 – podobieństwo do tych ostatnich sugerują kolorowe diody wskazujące aktywność urządzenia i odczyt danych z nośnika. Mnie osobiście bardzo przypadła do gustu – pomimo fatalnie umiejscowionych wyjść i wejść na peryferia. Jak dotąd jedynym mankamentem wykrytym przez ponad 17 lat użytkowania była pęknięta sprężyna dociskająca pokrywę napędu, która jak okazuje się, po wyjęciu jej z mechanizmu zamykającego, nie była niezbędna. Przez te wszystkie lata obudowa urządzenia nie zblakła ani nie pożółkła w odróżnieniu od jej starszych koleżanek. Siedmioprzyciskowy pad zachęcał i odstraszał jednocześnie. Bardzo wygodny, jednak ktoś nie pomyślał przy projektowaniu manipulatora kierunkowego. W krótkim czasie intensywna eksploatacja powoduje wyłamanie małych ząbków, w skutek czego „krzyżak" zaczyna się obracać, skutecznie uniemożliwiając granie. Podobnie dzieje się w przypadku gumowej nakładki manipulatora, która ma tendencję do odklejania się w najbardziej nieodpowiednim momencie rozgrywki.

Z czym to się je?

Wbrew wielu opiniom „ekspertów", konsola sama w sobie potrafi pokazać wiele. Napędzana jest przez praktycznie ten sam chipset, co jej większa siostra Amiga 1200. Posiada nowoczesny, 32-bitowy procesor Motoroli z serii 68k, czyli 680EC020 taktowany prawie 15 MHz – względnie tani procesor o wymiernych możliwościach, sprawdzony, z w miarę niewielkim poborem prądu, a co za tym idzie praktycznie znikomym wydzielaniem ciepła. Do tego chipset AGA mogący razić pełnym, 256-kolorowym obrazem, układ Paula, generujący ładny i przyjemny dla ucha czterokanałowy dźwięk o jakości znanej od lat wielu użytkownikom innych modeli spod znaku wielkiego C=. Dobrym rozwiązaniem było zastosowanie wewnątrz urządzenia pamięci podtrzymywanej „bateryjnie" – od tej chwili stany dłuższych gier mogły być zapisywane bezpośrednio, trwale, w przeznaczonym do tego celu układzie konsoli – dużo wygodniejsze rozwiązanie od wpisywania powszechnie używanych haseł i kodów. Z dodatkowych dobrodziejstw można wspomnieć o układzie zwanym AKIKO, który jest równie enigmatyczny, jak sama nazwa. Jedni mówią, że jest to chip wspomagający konwersję obrazu w formacie chunky do formatu planarnego, a inni upatrują w nim urządzenie sterujące CD-ROM-em. Trzeba wspomnieć, że w całkiem bogatej kolekcji tytułów wydanych na tę konsolę dosłownie jeden wymaga do uruchomienia właśnie tego wyspecjalizowanego układu. Na skutek powyższego istnienie tego „ustrojstwa" jest lekko zbędne, gdyż nie wspomaga on żadnego innego oprogramowania, o ile nie było pisane bezpośrednio pod niego. Co odróżnia tę konsolę od reszty dzieci wielkiego C= to brak stacji dyskietek! Zamiast niej wmontowany jest standardowy, całkiem niezłej jakości, jak na tamte czasy, napęd optyczny. Płytę urządzenia „wyzwolono" od wszelkich gniazd peryferyjnych Amigi 1200 dorzucając, obok złączy audiowizualnych, gniazdo o bardzo dobrych parametrach obrazu, czyli S-Video.

Upadek wielkiego brata

Wydaje mi się, że CD32 było deską ostatniego ratunku, jakiej chciało chwycić się chylące się ku upadkowi prosto na twarz, słynne Commodore. Przemawiać za taką tezą powinno to, że konstrukcja CD32 nie jest czymś nowym. Powstała na bazie istniejącej już architektury Amigi 1200 wskutek okrojenia płyty ze wszystkich części, czyniących z niej komputer domowy. Cięcie kosztów wymusiło zastosowanie starszego procesora masowo produkowanego dla siostry. CD32 korzysta także z tego samego, bootującego oprogramowania wzbogaconego jedynie o obsługę optycznego napędu. Innowacją miała być tajna broń – kość wspomagająca obliczenia. Obliczenia, które kosztowały Segę wprowadzenie przystawki 32X i dodatkowej stacji dokującej z napędem CD-ROM – kości specjalizowanej ładowanej do koszmarnie drogich kartów wielkiego „N" z serii FX. W CD32 mieliśmy to już na starcie. Możliwości 2D ówczesnych konsol 16-bitowych plus, w zamyśle konstruktorów, nowoczesne, 32-bitowe rozwiązanie w połączeniu z wygodnym odtwarzaniem animacji i muzyki prosto z nowomodnej płyty kompaktowej. Dodatkową opcją, niestety dostępną tylko dla wąskiej grupy posiadaczy CD32, było kupno trudno dostępnej, niezwykle drogiej przystawki wkładanej z tyłu urządzenia i zmieniającej komputer w rewolucyjny, domowy odtwarzacz filmów zapisanych w olśniewającej, jak na tamte czasy, technice cyfrowej formatu MPEG. Moduł jest jednym z produktów Commodore, który nie został należycie rozreklamowany. Dzięki temu rozszerzeniu konsola CD32 zamieniała się w pełnoprawny odtwarzacz filmów Video CD. Karta została oparta na układzie C-Cube CL450, potrafiącym dekodować w czasie rzeczywistym strumień MPEG (rozdzielczość 352x240 @ 30 Hz lub 352x288 @ 25 Hz) oraz wykonywać skalowanie zdekodowanego obrazu do rozdzielczości PAL lub NTSC. Z powodu wysokiej ceny i niewielkiej ilości sprzedanych egzemplarzy dziś jest to niezwykły i w dalszym ciągu kosztowny rarytas kolekcjonerski.

Konsola miała być chyba najtańszym rozwiązaniem, jaki sensownie było wypuścić na rynek w tym czasie. Ogromna ilość produkcji 16-bitowych działająca na maszynkach Commodore wymagała właśnie takiej architektury – dodatkowa pamięć, szybszy procesor i dodatkowe peryferia wydawały się być zbędnym i kosztownym wydatkiem. Może było to świadomym posunięciem w obawie przed niekompatybilnością klasycznego oprogramowania? Jak się okazało niebawem, nie było to do końca dobre, bo dużo konsumentów względnie szybko zaczęło narzekać na wtórność tytułów na platformie CD32 w porównaniu ze starszymi siostrami. Niestety marketing firm tworzących oprogramowanie działał trochę opacznie – bezmyślnie i głupio wydawano te same gry tylko na innym nośniku, bez dodatkowych wprowadzeń, animacji o muzyce odgrywanej ze ścieżek audio nie wspomnę. Sporadycznie zdarzały się nawet kuriozalne wersje okrojone. Ostatecznie, w ramach rekompensaty, wyeliminowanie z architektury stacji mało pojemnych nośników magnetycznych, dało rezultat wymierny w postaci wygodnego korzystania z oprogramowania podczas wesołego grania w domowych pieleszach.

Nowa konsola wytyczała też nową drogę, którą produkcja miała kroczyć w dziedzinie nowych projektów, którymi Commodore zamierzało zaatakować konsolowy rynek. Wymienić można tu projekt bardzo zgrabnej CD1200. Już na starcie miała być wyposażona we wszystkie dobrodziejstwa CD32 i być mniejszą gabarytowo o połowę od konsoli przystawką, umożliwiającą korzystanie z CD-ROM-u przez Amigę 1200, jak również dedykowanego tylko dla niej ekskluzywnego oprogramowania.

Brak oprogramowania

Drugą stroną medalu było to, że upadłe, w krótkim czasie po premierze CD32, Commodore nie zostawiło żadnego zabezpieczenia rynkowego dla ich najmłodszego dziecka. W efekcie bezlitosny marketing wymusił zamknięcie projektów dla sprzętu, który właśnie zaczął się całkiem dobrze sprzedawać. Niepewni swojego bytu, wystraszeni, ba – sterroryzowani faktem bankructwa firmy wiodącej od lat

prym w produkcji (tak, celowo użyłem tego określenia, bo mistrzem sprzedaży sprzętu C= to nie było, oj nie...) nowoczesnych, domowych maszyn komputerowych, producenci masowo kończyli pracę nad grami lub jak najmniejszym kosztem wprowadzali swoje ostatnie produkcje na szybko pustoszejącym rynku. Sytuacja ta napędziła niezłego stracha użytkownikom, którzy w obawie przed zamknięciem się na dopływ nowych produkcji, przesiadali się, na co popadnie – a raczej, na co było jeszcze kogo stać. Niektórzy zainteresowali się Megadrive, kolejni Super Nintendo, a następni poczekali i kupili Sony Playstation.

Poszerzanie horyzontów

Konsola doczekała się kilku ciekawych (dosyć egzotycznych) urządzeń rozszerzających jej możliwości, co okazało się działaniem niestety krótkotrwałym, bo najlepsze i najdroższe opcje balansowały na poziomie średniej dopalonej mocy oferowanej w tym czasie przez starsze, szybko taniejące siostry. W gruncie rzeczy, za ogromne pieniądze, dostawaliśmy niewiele więcej niż gołą Amiga 1200.

Płyta Amigi CD32 została wyposażona w 182-pinowy slot rozszerzeń. Choć była to przede wszystkim konsola, to jeszcze za życia Commodore prowadzone były prace nad kartą turbo przeznaczoną do tego slotu pod nazwą „CD/Game 030" (zachował się jedynie jej wczesny prototyp). Szczęściem w nieszczęściu było wprowadzenie na rynek, już po upadku Commodore, kilku przystawek produkowanych przez inne firmy. Najbardziej popularnym w naszym kraju rozszerzeniem było Elsat ProModule. Firma Elsat w momencie jej premiery była już znana jako producent akcesoriów do Amigi. Moduł ten rozszerza CD32 o brakujące elementy A1200, dzięki czemu konsola staje się pełnoprawnym komputerem. Firmie Microbotics udało się opracować rozszerzenie Microbotics SX1 – funkcjonalnie bardzo podobne do ProModule. Popularność tego modułu przyczyniła się do tego, że w późniejszym okresie rozszerzenia SX1 były produkowane przez firmy Paravision oraz Hi-Tech. Również produkty firmy DCE były oceniane bardzo wysoko. Dwa rozszerzenia do CD32 stanowią tego doskonały przykład: DCE SX32 i wersja Pro. SX-32 Pro jest najbardziej poszukiwaną przystawką a właściwie kartą turbo do CD32. Dzięki niej możliwe jest rozszerzenie konsoli o procesor 68030 oraz wszystkie dobrodziejstwa, które daje uboższa wersja SX-32. Wersja Pro dostępna była z procesorem taktowanym 40 MHz (bez MMU) oraz 50 MHz (z MMU). Dla Amigi CD32 pojawiło się również kilka mniejszych sprzętowych gadżetów, w których produkcji specjalizowała się warszawska firma Toms. Były to między innymi: rozdzielacz do gniazda AUX – NET FOR CD32 (umożliwiający komunikację pomiędzy CD32 a inną Amigą za pośrednictwem złącz AUX i Serial), interfejs klawiatury, dzięki któremu można było podłączyć tanią i ogólnodostępną klawiaturę od PC. Konsolę można także wyposażyć w scandoubler Indivision AGA – jest to najlepsze rozszerzenie graficzne do CD32 oferujące możliwość podłączenia konsoli do nowoczesnych monitorów LCD, krystaliczną jakość obrazu oraz wsparcie dla trybów HighGFX (co pozwala m. in. na używanie rozdzielczości 1024 x 768). W przeciwieństwie do opisywanych wcześniej rozszerzeń, nie jest on podłączany do slotu z tyłu CD32, lecz nakładany podstawką PLCC na układ Lisa. Niestety jest on dość trudno osiągalny.

Model przeznaczony do A4000D i CD32 został aktualnie wyprodukowany tylko w niewielkiej serii, która bardzo szybko się rozeszła.

Osobiście zawsze ciekawił mnie fakt braku zainteresowania rozszerzaniem sprzętowym możliwości płyty CD32 równocześnie z postępem prac nad „dopałkami" modelu A1200. Płyta ciemnoszarej przyjaciółki była przecież dużo młodszą konstrukcją – gabarytowo mniejszą, opcjonalnie tańszą alternatywą. Pomimo popularności i dosyć dobrej pozycji na terenie Wielkiej Brytanii, dziwi mocno fakt całkowitego braku zainteresowania podtrzymaniem projektu CD32 przez kolejnych właścicieli przejmujących prawa do maszyn i oprogramowania wielkiego C=. Dzięki temu bytność konsoli rozpływa się we mgle w ciągu 3-4 lat – niczym yeti w gęstwinie lasu. Zastanawiające jest też, że w niedługim czasie rozwiązania bliźniaczo podobne, czasem lekko ulepszone, jako rewolucyjne, znajdują zastosowanie w maszynach nowszej generacji i wywołają szeroki poklask.

Biblioteka oprogramowania

Dla Amigi pojawiło się wiele wspaniałych kompilacji wydanych na płytach CD – zbiory ciekawych zdjęć, produkcji scenowych, muzyki, modułów, wszelakiej maści programy. Niestety większość z nich do odpalenia potrzebowała systemu, którego na CD32 bez rozszerzenia ciężko było szukać. Na ratunek przyszła nam m. in. popularna kompilacja pod nazwą „NETWORK CD". Wystarczyło włożyć płytkę do konsoli, aby po chwili pojawił się znany nam Workbench. Bardzo ważnym i istotnym dodatkiem było wyłączenie funkcji reset związanej z otwieraniem czytnika. Dzięki temu rozwiązaniu właściciele nierozbudowanych konsol mogli korzystać ze wszystkich płyt wychodzących na Amigi i potrzebujących do odpalenia systemu, ponieważ bezproblemowo można było zamieniać płyty, a system z NETWORK CD odczytywał zawartość z innych krążków.

Jednym z najciekawszych, a zarazem nielicznych programów dedykowanych wyłącznie CD32 jest „Video Creator" niezwykle zasłużonej i nieodżałowanej firmy Almathera. Program pozwala tworzyć prezentacje multimedialne podobne do tych, które można tworzyć na znanej wszystkim amigowcom Scali. Można śmiało powiedzieć, że ostatnie dziecko Commodore doczekało się swojej wersji Scali, która oprócz standardowych efektów poznanych w innych tego typu programach, miała wiele własnych i nowoczesnych, jak na tamte lata, rozwiązań. Na przykład efekt overlay pozwalający na łączenie kilku obrazków ze sobą bez względu na to czy są one ruchome, dwu- czy trójwymiarowe. Program może być sterowany za pomocą pada lub myszki – mamy też możliwość korzystania z muzyki odtwarzanej z kompaktu i długich animacji.

CD32 posiada liczną bibliotekę gier, której bliżej przyjrzymy się w kolejnym odcinku zamieszczonym w następnym numerze magazynu. Teraz tylko wspomnę, że „znawcy" tematu zazwyczaj wskazują tylko kilka z nich, jako tych godnych uwagi. Resztę krytykuje się za wtórność z wersjami dyskietkowymi. Opuścimy na to zasłonę milczenia i zajmiemy się całościowo zbiorem wszystkich wydanych na tę platformę produkcji podzielonych na kilka kategorii. Ale o tym w kolejnym numerze...

Doceniona dopiero po latach

Przegrany, pokonany czy może raczej niedoceniony gracz? To ostatnie raczej. Na skutek wszystkich wydarzeń związanych z upadkiem Commodore, konsola na wiele lat popadła w niełaskę, ale czy słusznie? Chyba nie. CD32 to wciąż łakomy kąsek dla każdego wielbiciela gier wideo i konsolowej rozrywki. Okazuje się, że nadal do dużej części klasycznych gier na maszyny Commodore wcale nie potrzeba „wypasionych" konfiguracji.

Małe, niepozorne, szare pudełko i wiele kolorowych gier – sporo osób błędnie zarzuca mu niewielkie możliwości. Goła konsola już na starcie oferuje wszystko, co niezbędne do sensownego odpalenia 90% oprogramowania aktualnie dostępnego w chwili jej wypuszczenia na rynek. W prostej linii tanim kosztem można było zalewać rynek odświeżonymi, 16-bitowymi tytułami, które zdobyły już status klasycznych lub kultowych. Możliwości obliczeniowe na to w zupełności pozwalały. Układ

Dokończenie na stronie 27

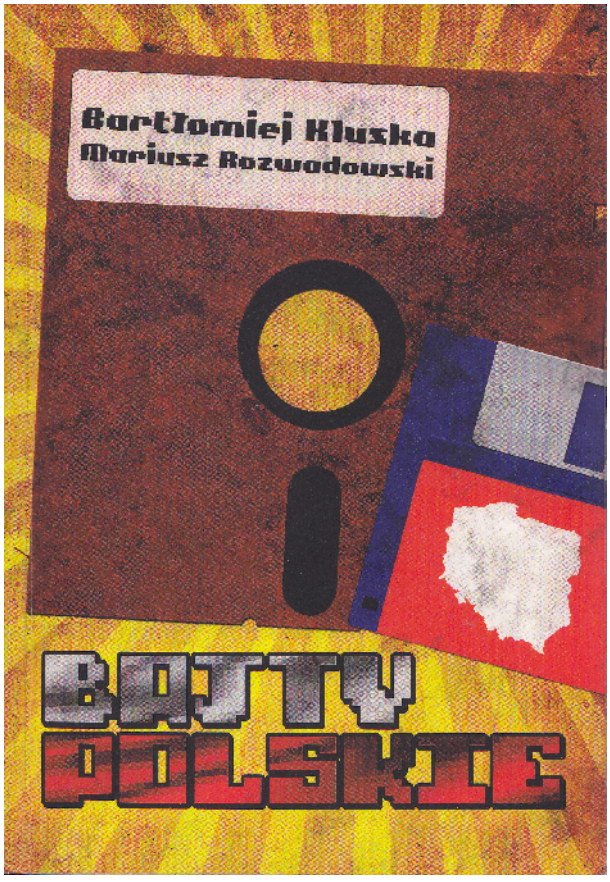
„Bajty polskie” – recenzja

Zawsze unikałem pisania recenzji jakiejkolwiek książki, żeby nie być posądzonym o sprzyjanie autorowi za pieniądze lub robienie mu złej opinii za pieniądze konkurencji. Rynek książki to rynek trudny i tu dużo tego typu rzeczy ma niestety miejsce. Czegóż to się nie robi, żeby sprzedać produkt (w tym wypadku książkę), który się wyprodukowało. Tak więc musicie mi uwierzyć na słowo, że autorów *„Bajtów polskich”* nie znam osobiście ani nic mnie z nimi nie łączy. Po co taka asekuracja już na samym początku? Ano dlatego, że książka *„Bajty polskie”* to pozycja wyjątkowa i podczas całej niniejszej recenzji będę bronił tej tezy jak lew, co niektórym zwolennikom spiskowej teorii dziejów może się wydać cokolwiek podejrzane.

Jak ja nie mogłem się doczekać przyjścia przesyłki. Długo to nie trwało, ale dłużyło się strasznie. Wiedziałem, co mnie czeka. Po przeczytaniu poprzedniej książki Bartłomieja Kluski *„Dawno temu w grach”* wiedziałem już, że będę miał do czynienia z dziełem wyjątkowym. Próba opisania historii rozrywki elektronicznej to nie była pierwsza tego typu próba w Polsce, ale to była pierwsza udana próba. Tak więc przesyłka przyszła.

Książka *„Bajty polskie”* Bartłomieja Kluski i Mariusza Rozwadowskiego ma 213 stron formatu A5 i jest oprawiona miękką okładką raczej odporną na zabrudzenia. Druk jest czarno–biały, nie ma kolorowych wkładek na zrzuty z ekranów gier, jak to zostało zrobione w *„Biblii komputerowego gracza”* Aleksego Uchańskiego. Trochę szkoda, bo to byłby miły dodatek. Rozumiem jednak, że zdecydowały względy ekonomiczne i nie traktuję tego jako wadę książki, ale jako smutną konieczność wymuszoną realiami rynku. Tak więc całość jest czarno–biała, zrzuty z ekranów gier również. Autor zaczyna od zgrabnie napisanego wstępu (nazywając go *„Intro”*) by stopniowo wprowadzać czytelnika w świat polskich początków informatyki. Ponieważ podjadałem przez całą noc, przez którą przeczytałem książkę całą (trudno się oderwać od lektury), to jedyne co mi przychodzi do głowy to stwierdzenie, że książka jest smaczna. Znakomity styl autorów powoduje, że nie czytamy encyklopedii, tylko razem z autorami przeżywamy początki informatyki w Polsce i związanej z nią rozrywki elektronicznej. Obraz tamtych czasów sprawia wrażenie kompletnego i spójnego. A do tego napisany jest zajmująco. Nie będziemy się jednak oszukiwać – książka jest dla tych, którzy choć trochę czują klimat pierwszych gier i czują magię tamtych czasów. Pokolenie wychowane na grafice 3D i grach online książka będzie nudzić niemiłosiernie. Nie zainteresuje ich również ze względów poznawczych – to jest książka tylko dla weteranów. Po suchą wiedzę mogą zajrzeć do Wikipedii. Ta książka jest soczysta i pełna emocji.

Autorzy nie pomijają opisów realiów rynku, a więc wszechobecnego piractwa, które tak naprawdę toczyło rynek po równi pochyłej. Nie unikają tematów trudnych związanych z tworzeniem gier, a więc permanentny brak narzędzi do ich tworzenia i brak polskich tłumaczeń do instrukcji obsługi. Aż dziw bierze, jak w takich trudnych warunkach mogło powstać tyle wspaniałych i kultowych gier. Autorzy podpisują całą swoją książką czasy, w których została ona napisana, czyli nam współczesne. Na każdym kroku jest twarde stąpanie po ziemi i przyglądanie się realiom ekonomicznym. W zestawieniu z tymi romantycznymi porywami tamtych czasów, wiarą, że na grze będzie można zarobić, ale że przede wszystkim będzie można zaistnieć na rynku, wygląda to niesamowicie. Choć zapewne to nie było zamierzeniem autorów, niechcący opisali świat fantazji i iluzji istniejący w głowach tylko tych młodych ludzi, całkowicie oderwany od realiów ekonomicznych. Pokazali, że to było piękne, a niekoniecznie opłacalne. Zapewne niechcący wyszedł obraz pokolenia „ulepionego z innej gliny”. Sam zresztą w 2002 roku wysmarowałem przydługiego lekko maila do wydawnictwa o tym, jak to w ogóle Axel Springer Polska mogło zamknąć po kilku numerach najlepszy magazyn o grach *„Top Secret”*. Miałem w tylnej części ciała realia ekonomiczne – ja chciałem mieć co miesiąc *„Top Secret”* w kiosku osiedlowym i koniec. I nawet dostałem odpowiedź (z tego, co pamiętam, odpisał Sir Haszak), że *„pismo przynosi straty finansowe i wydawca zdecydował się na zamknięcie pisma”*. Twarda ekonomia – koniec.



Każdy z zainteresowanych, nieważne, jaki miał komputer w tamtych czasach lub nadal ma w tych, znajdzie coś dla siebie. Jednak Amidze i pecetowi, wiecznie ze sobą wtedy konkurującymi maszynami, poświęcone zostały oddzielne rozdziały *„Bez kompleksów: AMIGA”* i *„Bez kompleksów: PECET”*. Na oddzielny dział zapracowało sobie również Atari. Z racji tematyki niniejszego pisma zajmę się oczywiście Amigą. Środowisko amigowe przedstawione jest w tym rozdziale jako bardzo ambitne, choć „nie zawsze szły za nimi umiejętności...” Tu tak, jak w każdym środowisku, poruszone są dwa podstawowe problemy w dużej mierze powiązane ze sobą: brak pieniędzy na tworzenie gier i wszechobecne piractwo. Mimo to Polacy mają się czym poszczycić. Wymieniona została chyba większość ważnych gier amigowych „made in Poland”. I tu taka mała uwaga ode mnie: ja bym obok przygodówek takich jak *„Mentor”*, *„Eksperyment Delfin”* czy *„Misja Harolda”* jednak wspomniał o znakomitej grze *„Alfabet Śmierci”*. O firmie Seven Stars, wydawcy *„Alfabetu Śmierci”* autorzy wspominają przy okazji gry *„Kajko i Kokosz”*. Dlatego tym bardziej dziwi mnie, że ta gra została pominięta - w moim odczuciu jest znakomita, choć niektórzy mogą uważać inaczej. Na kartach traktujących o Amidze nie zostały pominięte również zakały rynku i gnioty. Firma L.K. Avalon, która robiła znakomite gry na „małe” Atari, już tak dobrze nie sprawdzała się na rynku „przyjaciółki”. Ja bym do wymienionych gier *„Noc”*, *„Rajd przez Polskę”*, *„Sen”* dorzucił jeszcze *„Tajemnicę Bursztynowej Komnaty”* i *„Łowcę głów”*, a dla miłośników gry *„Superfrog”* wspomniałbym jeszcze grę *„Forest Dumb”*. Zaszczytne miejsce w tym akapicie zajmuje również firma Techland. Koniec ery Amigi jest przedstawiony jako ten moment, gdy Amiga nie daje sobie rady z rewolucją 3D, czyli pojawieniem się gry *„Doom”*. Oczywiście autorzy nie pomijają licznych i wielu udanych klonów tej kultowej gry, ale to był początek końca Amigi na wielkim rynku – tak autorzy umiejscawiają to w czasie, choć jak piszą: *„lecz amigowcy, zanim ich sprzęt odejdzie do lamusa historii, mieli światu jeszcze coś do udowodnienia. Musieli zrobić amigowego „Dooma””*.

Autorzy nie zapominają o młodszych siostrach komputerów – konsolach. I tu też sama pragmatyczna rzeczywistość. W tym kraju, w tych pionierskich czasach była obecna tylko jedna konsola – podrobiony NES, zwany popularnie Pegasusem. I co z tego, że praktycznie każda konsola do gier była od tego śmiesznego pudełka, zwanego szumnie *„Family video computer”* lub *„Family video game”* (w zależności co tam nakleili na obudowę), lepsza skoro Pegasus był pierwszy. Autorzy nie zapominają, że to też była elektroniczna rozrywka (od której na przykład ja zacząłem swoją przygodę z tym dziwnym

światem). Jak już wspomniałem, autorzy nie omijają tematów trudnych i niewygodnych oraz osobistych. Tak więc zagadką pozostaje fakt, dlaczego Janusz Pelc *„konsekwentnie odmawia udzielania jakichkolwiek informacji na temat własnej przeszłości z czasów, gdy zajmował się pisaniem gier komputerowych”*. Panie Januszu, nie ma się czego wstydzić. Jest pan ikoną „małego” Atari.

Treść książki, jak już napisałem wcześniej, jest soczysta. Ale autorzy, jakby tego było jeszcze mało, nie poprzestają tylko na treści, ale idą o krok dalej i prezentują tę treść w bardzo atrakcyjnej formie. Przez całą książkę przewijają się ciekawostki z danego tematu, oplecione ramką z tytułem *„EXTRA DATA”*. Ramka stylizowana jest na okno systemu operacyjnego. Nie dam sobie nic uciąć, ale to chyba amigowy Workbench. Będzie dziwnie i strasznie, a także tajemniczo. Niektóre informacje w tych ramkach naprawdę mogą człowieka mocno zdziwić. Zresztą kilka recenzji tejże książki wspomina o tych smaczkach. Wydawało mi się, że dość dobrze orientuję się w temacie tamtych czasów, ale tylko mi się wydawało. Autorzy dotarli do takich informacji, których nie ma nigdzie indziej! Nigdzie. Jak kupicie książkę, to sami zobaczycie. Co do ciekawostek, autorzy wspominają o dziełach niedokończonych lub nigdy nie wydanych z różnych powodów. Naprawdę nie chce psuć pierwszego wrażenia po przeczytaniu paru ciekawostek. Zachęcam do lektury.

Autorzy, przymierzając się do kończenia książki, zaznaczają wyraźnie, gdzie kończy się jedno pokolenie graczy, a zaczyna kolejne. Podział jest czytelny i jaskrawy. I trudno się nie zgodzić. Nie wymienię tej granicy wprost, ale zacytuję autorów *„... utrwalona w nazwie magazynu płyta kompaktowa, na której jeszcze w 1996 roku zaczęły pojawiać się pełne wersje gier, okazała się rynkowym hitem. Mimo rezerwy, z jaką debiutujący miesięcznik przyjęli weterani grania na komputerze, nowe pokolenie graczy potrzebowało nowego czasopisma...”* oraz kończące praktycznie ostatni rozdział książki słowa: *„Jest to już jednak zupełnie inna historia”* (potem są tylko podziękowania i...) mówią same za siebie. Moim zdaniem to bardzo dobrze, że autorzy jasno podkreślili tę granicę, ten fakt dla mnie osobiście smutny, że pewna epoka się skończyła i już nigdy się nie powtórzy. Dobrze, że postawili czytelną granicę między tamtym a tym pokoleniem.

Jak każda rzecz, która sprawia nam przyjemność, niedobrze i źle się czujemy, że się w końcu kończy. Ta książka też się kończy z optymistycznym tytułem rozdziału *„Ciąg dalszy nastąpi”*. Autorzy zapowiadają w nim kolejną, niezwykłą książkę, jakby jeszcze po lekturze tej znakomitej książki pozostawał jakiś niedosyt. Otóż zapowiadają książkę, *„która będzie zawierała wykaz absolutnie wszystkich polskich gier komputerowych, opisy i ilustracje, fragmenty recenzji, „kody na nieśmiertelność”, jak również wypowiedzi ich twórców i wydawców, anegdoty, mało znane fakty oraz inne atrakcje.”*

Napiszę krótko: jako niewierny Tomasz, nie uwierzę, póki nie zobaczę! To byłoby (czy też będzie, jak zarzekają się autorzy) wydarzenie bez precedensu na skalę chyba światową. Oczywiście do tego opasłego tomiska, rozpieszczony dwoma poprzednimi tytułami, spodziewałbym się jako czytelnik jakiejś wisienki na torcie, jakiegoś „bonusa”. Widziałbym tu płytę CD lub DVD z prezentacją każdej z tych gier, lub chociaż zrzutów z ekranu! Jako dodatek taka srebrzysta płytka ładnie wydana sprawdziłaby się doskonale. Czy realia ekonomiczne pozwolą na zrealizowanie tego pomysłu? A może potrzeba tutaj tego zapału i romantyczności pierwszego pokolenia? Zastanówcie się nad tym drodzy autorzy.

Podsumowując: książka jest znakomitą i obowiązkową pozycją każdego fana rozrywki elektronicznej w jakiejkolwiek postaci. Tę książkę po prostu trzeba mieć. I na końcu apel – tę znakomitą książkę na pewno będzie można pobrać z sieci za jakiś czas w postaci przeskanowanego mniej lub bardziej niechlujnie PDF-a. Nie pobierajcie tego. Kupcie książkę od autorów, zapłaćcie im za trud, jaki włożyli w stworzenie tego małego dzieła sztuki. Jeśli ukradniecie książkę, następna może się już nie ukazać i podzieli los wielu gier, jakie miały się ukazać, ale autorzy zwątpili w sens ich dokończenia.

Książkę można zamówić w wydawnictwie Orka (*http://www.orka.media.pl*)

Tomasz Dębkowski

CD32

Dokończenie strony 25

Akiko mógł z powodzeniem uciągnąć królujące w tym czasie gry na innych platformach, takich jak Sega 32X czy Nintendo SNES z chipem FX. W perspektywie opracowanego już na starcie rozszerzenia z potężniejszym procesorem, kto wie czy sprzęt nie mógłby konkurować później ze słynnym Saturnem lub PSX. Zastosowane złącze S-Video znalazło uznanie dopiero wiele lat później! To wskazuje, jak wizjonerskie było jego zastosowanie. Po dziś dzień znane są zabiegi, kiedy to demontuje się ten cały układ z płyty CD32, by przylutować go do bebechów Amigi 1200 w celu uzyskania wysokiej jakości obrazu generowanego przez zastosowane w obydwu modelach układy graficzne AGA!

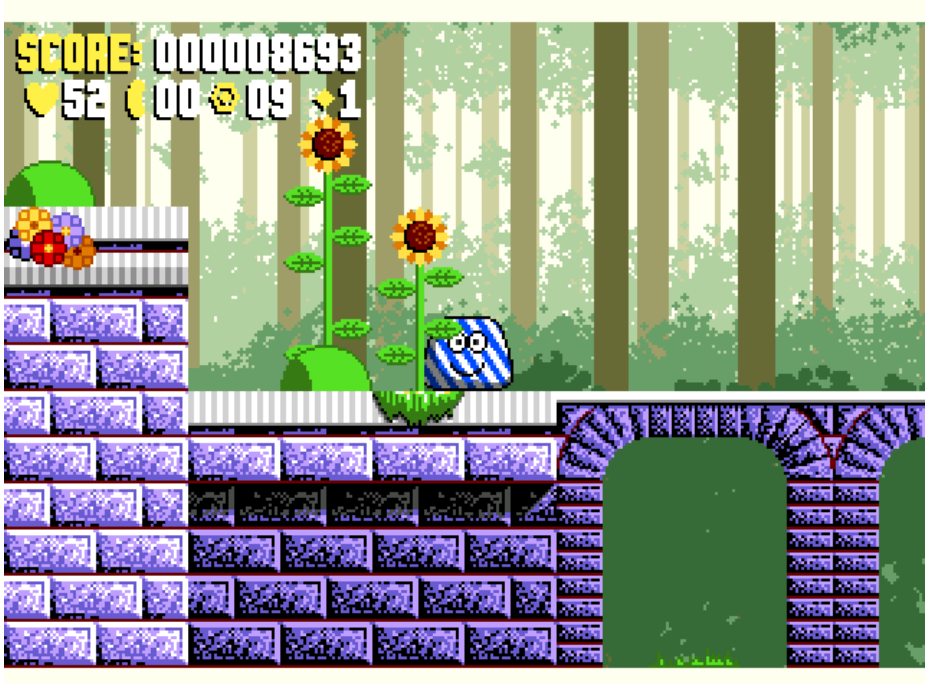
Napęd optyczny i nośnik CD-ROM przez wiele lat był najtańszym i powszechnie używanym standardem. Kolejne generacje płyty CD wskazują na to, że jeszcze przez wiele lat „kompakt” będzie najpopularniejszym nośnikiem w obiegu. Rosnące wciąż zainteresowanie klasycznymi grami, unikalnym sprzętem lat 90-tych sprawiło, że konsola staje się niezłym i niestety coraz droższym kąskiem kolekcjonerskim. Sprawne zestawy wraz z oryginalnymi manipulatorami i choć niewielkim zestawem oryginalnych gier, należą do kosztownej rzadkości. Dużym plusem sprzętu jest całkowicie otwarta architektura konsoli, pozwalająca na uruchamianie na niej przygotowanych przez siebie, domowym sposobem, płyt CD. Za pomocą prostych kompilacji setki gier, które nigdy nie zdążyły ukazać się na CD32 uruchamiają się i nadal dają wiele radości!

Malejąca dostępność nośników magnetycznych i powolne starzenie się aktualnie używanych, spowodowało chęć stworzenia pakietów przenoszących nawet trudne do zainstalowania na twardym dysku gry i programy. Nagranie wszystkiego na samobootującej się płycie zaczęło mieć sens. Napęd optyczny oferuje sporo miejsca. Dostęp do danych jest względnie szybki – wystarczy, jak przed laty, włożyć nośnik do napędu i już po chwili można cieszyć się wspaniałymi kolorowymi grami, tym co pozostało po wspaniałych firmach, programistach, grafikach i muzykach działających blisko dwie dekady temu!

Radosław „Strim” Kujawa
Łukasz „Vegeta” Jeglorz
Krzysztof „Koyot1222” Matys

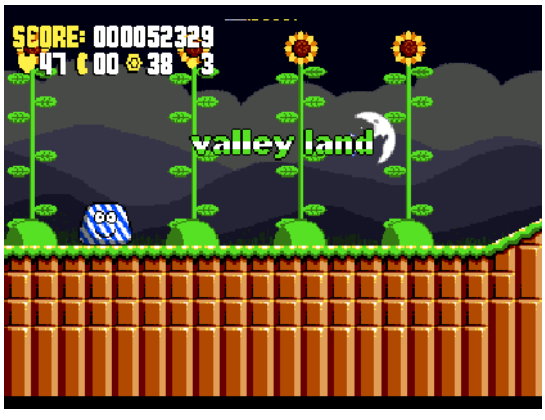
Mr Beanbag!

„Mr Beanbag!" to gra zaliczana do dosyć popularnego gatunku platformówek. Wiele osób zapewne zakrzyknie „kolejna?!", lecz trzeba odnotować, że w tej swojej powtarzalności cechuje ją coś nietypowego. Po pierwsze, gra jest nowa – wydana całkiem niedawno po prawie siedemnastu latach prac. Początkowo pisano ją w AMOS-ie, lecz z czasem przepisano wszystko w assemblerze (miało to miejsce w 2005 roku) i w tym języku grę ukończono. Po drugie, co jest dosyć nietypowe w amigowym światku, autorem jest kobieta – Szkotka o scenowej ksywie Tricky, a prawdziwym imieniu Amy Worthington. Po trzecie, gra jest darmowa. A po czwarte i po piąte...



„Mr Beanbag!" pracuje wyłącznie na komputerach wyposażonych w chipset AGA, tak więc już z założenia pewna część odbiorców zostaje wykluczona z zabawy. Czy obecnie ma to jakieś znaczenie? Czy może to oznaczać jakąś komercyjną porażkę? Nie sądzę, gdyż chyba nie w tym cały sens. Gra charakteryzuje się bardzo szybkim przesuwem ekranu, co ma niesamowite walory rozrywkowe. Zadaniem gracza jest sterowanie główną postacią i pokonanie ponad czterdziestu etapów (w dziesięciu światach) napakowanych różnymi przeszkadzajkami w celu uratowania niejakiego pana Fizzy Popa z rąk hrabiego Gazpacho. Nasz bohater, kształtem przypominający torbę lub podusię, nie posiada żadnej broni. Eksterminuje przeciwników, naskakując na nich lub wprowadzając siebie w ruch obrotowy, co jest zgubne w skutkach dla wszystkiego, co znajduje się na jego drodze (z wyjątkiem kolców). Wydawać by się mogło, że obracanie może stać się istotą gry, lecz aby móc się obracać, należy najpierw odpowiednio się rozpędzić. Na drodze naszej wycieczki znajdować będziemy słodycze, które należy wcinać, tym samym zwiększając nasz dobytek punktowy. Niektóre ze znalezionych fantów pozwolą nam uzupełnić nadszarpnięte siły witalne, a i nawet zwiększyć liczbę wcieleń.

Istotnym elementem podróży jest nieliniowość rozgrywki. Grę rozpoczynamy w rodzinnej dolinie naszego bohatera, po czym kolejność, w jakiej będziemy odwiedzać kolejne krainy, zależy już wyłącznie od naszych upodobań. Wyboru etapu dokonujemy na mapie, przy czym każdy już odwiedzony możemy eksplorować dowolną ilość razy. Po niepowodzeniu gra oferuje nam możliwość rozpoczęcia zabawy od jednego z ostatnich miejsc, które możemy wskazać przy pomocy kodu dostępu (uwaga – kody dostępne są wyłącznie z poziomu głównego menu).



Graficznie gra wykonana jest dosyć ascetycznie. Nie mam tutaj na myśli jakieś okropnej grafiki przypominającej trzy kreski i kółko. Grafika w grze jest po prostu inna niż wszystko, dosyć nietypowa. Nie była rysowana przez zawodowego grafika, lecz najzwyczajniej w świecie przez osobę, która ma zmysł kolorystyczny i nic więcej. Ale to wystarczyło, aby stworzyć coś intrygującego i ładnego, bez zbędnego przepychu. Bardzo trudno jest to opisać, ale wystarczy popatrzeć na obrazki. To jest tak proste, że aż ładne i szkoda, że w ten właśnie sposób, w dobrych amigowych czasach, nie robiło się grafiki do wielu ohydnie wykonanych produkcji. W kwestiach graficznych przyczepiłbym się jednak do momentami zbyt wielu pustych przestrzeni na ekranie. Co prawda nadużyciem jest podwyższenie poprzeczki aż do poziomu chipsetu AGA, gdyż grafikę zastosowaną w grze spokojnie uciągnęłaby wysłużona A500 (a i nawet widzieliśmy na niej lepszą) – po prostu nie ma tutaj elementów tętniących tysiącami barw. Ciekawie wygląda efekt paralaksy zastosowany na tłach, a jeszcze większą ciekawostką jest to, że owe tła nie są miernej jakości gradientami kolorów, lecz obrazkami w stonowanych odcieniach. Dodajmy do tego element zwierciadła wody w niektórych etapach i chyba powoli zaczynamy rozumieć, dlaczego gra wykorzystuje chipset AGA, choć nadal nie jest to dostateczne usprawiedliwienie. Jak dla mnie, dla A500 grafika trzyma poziom dobrej platformówki, ale dla A1200 jest to nadal stanowczo za mało (choć bynajmniej nie można tego traktować jako błąd). Muzycznie gra trzyma poziom. Utwory towarzyszące rozgrywce są miłe dla ucha. Zostały napisane przez autorkę – nie nudzą ani nie męczą. Są w sam raz i pozwalają, każdemu fanowi gier platformowych zatopić się w klimacie gry.

„Mr Beanbag!" to gra z pozoru bardzo prosta. Jedna z wielu, jakie na Amidze się pojawiły i jednocześnie jedna z wielu, w które nie było dane nam zagrać, bo uznaliśmy, że gatunek, który reprezentuje, jest zbyt wyeksploatowany. Z tym stwierdzeniem można się zarówno zgodzić, jak i nie zaakceptować. Nie chcę nikomu narzucać zdania odnośnie tej gry, ale zachęcam do wypróbowania i przekonania się na własnej skórze. Sądzę, że wielu z Was będzie mile zaskoczonych, że nawet teraz można stworzyć coś ciekawego i w odpowiedniej jakości. Jakości, która jest czymś więcej niż tylko akceptowalna. Szybko się przekonacie, że pogoń przed siebie na oślep jest zgubna w skutkach. Etapy zostały pomysłowo zaprojektowane i potrzeba poświęcić trochę czasu na ich dokładne zbadanie. Dochodzi do tego również pomysłowość i typowe podejście zrobienia gry takiej, która nie będzie zawierała niedociągnięć, jak nasi ulubieńcy z przeszłości.

Gdy będziecie sięgać po *„Mr Beanbag!"* pamiętajcie, że grę stworzyła praktycznie jedna osoba z niewielką pomocą swoich przyjaciół, którzy służyli radą, pomagali w testach i projektowaniu etapów. Co więcej, stworzyła ją przedstawicielka płci pięknej, co nie należy do częstych obrazków, zwłaszcza na naszym poletku. Patrząc przez ten pryzmat, porównajcie ją z wielkimi, amigowymi hitami z lat 90-tych wydanymi przez firmy zrzeszające programistów oraz z zalewającymi amigowy rynek produkcjami, które nie cieszyły się popularnością, a jednak je ktoś wydawał, licząc na zasobne portfele „amigantów". Mając to na względzie, szybko dojdziecie do wniosku, że *„Mr Beanbag!"* to naprawdę łakomy kąsek. Zwłaszcza teraz – gdy w zasobniku gier straszna posucha, a propozycja przemiłej Szkotki jest na pewno czymś więcej niż przysłowiowym rakiem na bezrybiu.

Sebastian Rosa