Polskie Pismo Amigowe

Numer 3/2014 (13)

cena 20 zł

Workbench release. 852440 free memory

Prefs

Rise of the Triad

Goatlizard DX

RAM DISK

Workbench1.3

Redit

Open X.Com

ISSN 2354-0273

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:

15 listopada 2014 roku.

Zapraszamy do udziału!

Polskie Pismo Amigowe

Redaktor naczelny: Sebastian Rosa

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

Współpracują: Mateusz Eckert, Aleksander Giedyk, Tomasz Pacyna

Skład: Sebastian Rosa

Grafika: Sławomir Woźniak

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

Skład pisma wykonywany jest w programie OpenOffice 3.

Druk: Drukarnia cyfrowa DCTF w Tarnowie.

Wydawnictwo AIBB – Tomasz Bernacik
ul. Piotrkowska 201 lok.10, 90-451 Łódź

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.

http://www.ppa.pl

W numerze

- **Furia EC020** 4
- **BetterWB czy ClassicWB** 6
- **MomosIRC** 9
- **AmiKit 7 – na gorąco** 10
- **Redit** 12
- **2048** 13
- **Open XCom** 14
- **Nasza pierwsza gra** – kurs programowania pod AmigaOS – część 8 16
- **Zabawa obrazem w Pythonie** 18
- **Rise of the Triad** 20
- **PT-1210 MK1 Protracker Digital Turntable** 21
- **Goatlizard DX** 22
- **Jedi Knight: Jedi Academy** 24
- **Scenowe graffiti** 26

Od redakcji

Często sięgam do starych amigowych pism, aby chwilę powspominać czasy młodości, gdy z wypiekami na twarzy biegłem do kiosku lub na giełdę komputerową po nowy numer moich ulubionych periodyków. Zapach farby drukarskiej oraz szelest przewracanych kartek do dzisiaj rodzą niezapomniane wspomnienia. Podczas tej mojej wycieczki w czasie, dopada mnie refleksja czy obecnie młodzież, a i nawet my – stare, komputerowe wygi – choć przez krótką chwilę doświadczamy tego samego. Oczywiście technologia jest trochę inna i nie wiem jak u innych, ale dla mnie oczekiwania i wrażenia są dokładnie takie same. Gdy przychodzi do mnie koperta z nowym numerem *Amiga Future* lub *Retro Gamerem*, za każdym razem z taką samą fascynacją i zainteresowaniem przeglądam kolejne strony, zaczynam lekturę, którą kończę tuż przed premierą następnego numeru, na który zawsze czekam z niecierpliwością domyślając się, co też tym razem w nim przeczytam. Obie lektury wzajemnie się uzupełniają – *Retro Gamer* zapewnia mi powrót do przeszłości, wspomnienia, ciekawe historie, a *Amiga Future*, choć lingwistycznie raczej słabe, daje lekki powiew świeżości rzucając okiem na amigowe nowinki.

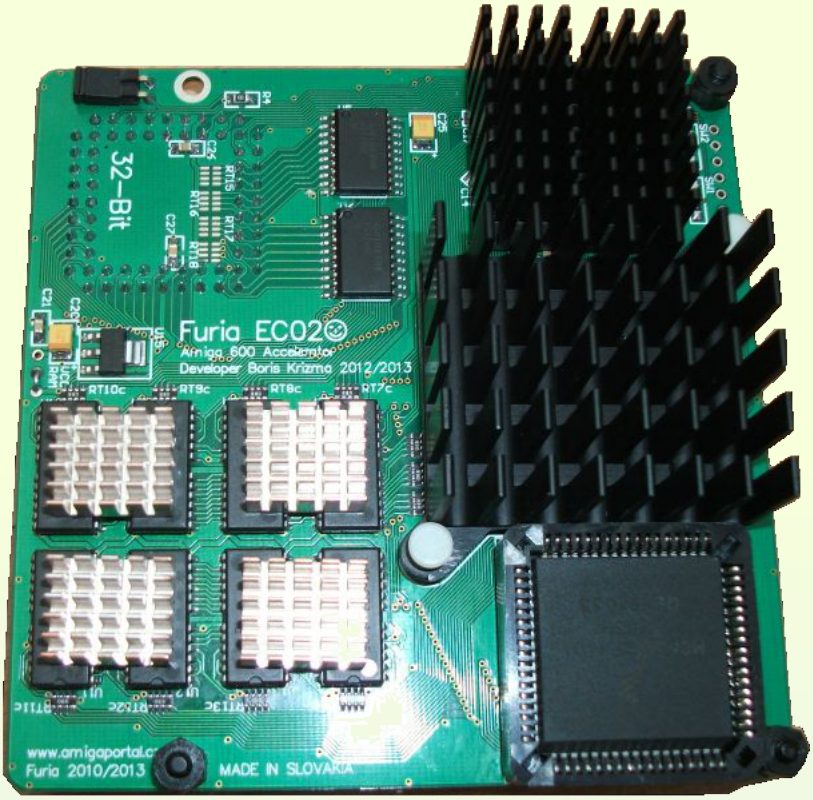
Tak co miesiąc dopada mnie także jeszcze jedno – dlaczego coś takiego nie jest możliwe po polsku? Trochę zaczepnie postawione pytanie, bo przecież mamy *Polskie Pismo Amigowe*, którego kolejny numer możesz trzymać w rękach, drogi Czytelniku, jak również piszę te słowa tuż przed premierą *Amigazynu* oraz reaktywowanego *Secret Service*, o którego formie niewiele można powiedzieć. Tylko nieliczni jednak zdają sobie sprawę, jak ciężko w obecnych czasach, taki periodyk wydawać. Jak zażarcie trzeba prosić o to, aby znalazły się osoby zainteresowane zapełnieniem tych kilkunastu stron tekstu, aby inni raz na kwartał mogli na chwilę cofnąć się w czasie i potrzymać w dłoniach papier przesiąknięty amigową (chociaż może nie tylko) nutą. Cieszę się z tego, że udało się zbudować pewną kadrę redaktorów chętnych do napisania kilkunastu słów, dzięki którym możesz czytać to, co właśnie masz przed sobą, drogi czytelniku, ale jednocześnie ubolewam, że jest ich tak mało. Internet wielu z nas rozleniwił – wolimy pisać w puste na forach o tym, co potrafimy i co zrobimy, ale dopóki te przechwalanki nie zamienią się w namacalny byt nic nie znaczą i, co właśnie najgorsze, doprowadzają do upadku ciekawych inicjatyw. A tego nikt z nas chyba by nie chciał. Postarajmy się pokazać, że prasa tematyczna może żyć tak długo, jak są ludzie, którzy chcą ją czytać, a nie tak długo jak ludzie, którzy chcą ją pisać. Ci drudzy zazwyczaj znikają przed tymi pierwszymi... A co jest dalej, to już przerabialiśmy na tym ziemskim padole wiele razy.

Z tego miejsca chciałem podziękować tym, którym się chce zrobić coś więcej niż zapełniać bazy danych dysków twardych serwerów, na których stoją internetowe fora. Te osoby nie robią tego dla pieniędzy ani dla sławy – robią to właśnie dla Ciebie, aby takie refleksje, jak moje, nikogo nie dopadały i aby każdy z nas mógł choć przez krótką chwilę powspominać swoją młodość.

P.S. Tylną okładkę numeru stanowi JEDNA nadesłana praca na ogłoszony przez nas konkurs, o który zabiegali czytelnicy. Jej tytuł to *„Planeta Amiga"* i, jak napisał jej autor, *„stanowi analogię do naszego amigowego światka. Jest on bardzo mały, ale musi pomieścić ludzi o różnych poglądach, zwolenników różnych rozwiązań. Na tej planecie dla każdego z nich powinno znaleźć się miejsce."*

Sebastian Rosa

Furia EC020



Po powrocie do amigowania, na początku 2011 roku, nabyłem w sklepie eFunzine A1200 wraz z Blizzardem 1230 Mk III. Wówczas jeszcze nie przypuszczałem, że bakcyl, jakim się zaraziłem, stanie się poważną epidemią, na którą antidotum nie znalazłem do dnia dzisiejszego. Niedługo później, na znanym portalu aukcyjnym, wypatrzyłem fajny zestaw A600 z rozszerzeniem pamięci Chip do 2 MB. Rzecz była o tyle nietypowa, że model rozszerzenia znałem tylko ze słyszenia, a zaintrygował mnie ze względu na karty pamięci SRAM na PCMCIA dające dodatkowe 1, 2 i 4 MB pamięci Fast oraz obecność złącza IDE. Gdy go kupowałem na rynku były już dostępne karty ACA 630/25 MHz i 30 MHz, jednak po zakupie „tysiącdwusetki" byłem na tyle spłukany, że mogłem sobie pozwolić tylko na tyle. Brak funduszy był również pomniejszą przyczyną tego, że ominęła mnie pierwsza seria A6095 i tylko dzięki szybciej reakcji załapałem się na drugą edycję. Co by jednak nie pisać, doświadczenia zebrane z używania kart PCMCIA oraz A6095 spowodowały, że owszem – tak dopalonej A600 można było używać jako platformy do WHDLoad, jednak dla mnie gry to nie wszystko. Lubię coś porobić „na systemie", a w tych niestety przypadkach, jest to mało komfortowe i jak dla mnie niewystarczające.

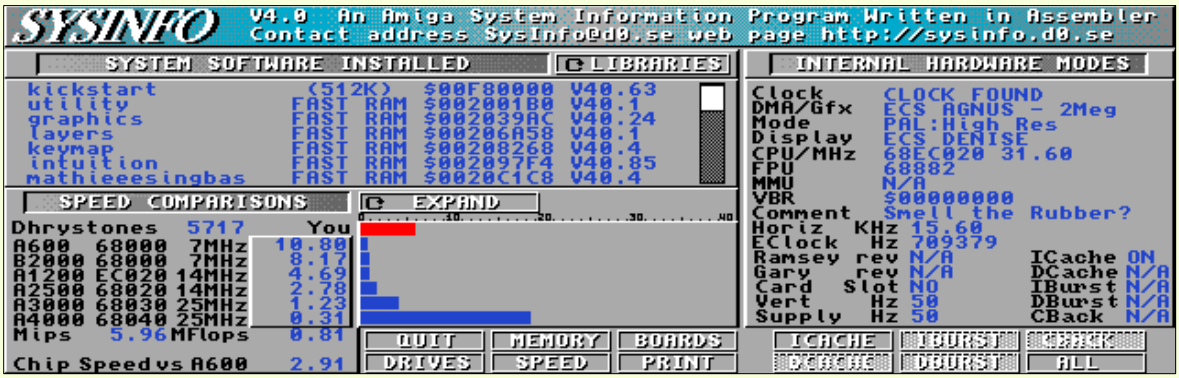
W sierpniu 2012 roku natrafiłem na naszym forum na informację o karcie Furia S628, której twórcą jest Boris „Boboo" Krizma. Kartę wyposażono w procesor 68000 28 MHz oraz 2 MB pamięci Fast. Przyspieszenie ponad 5 razy względem fabrycznej A600 robi wrażenie, choć ilość pamięci wywołuje pewien niedosyt. Z tego też powodu zrezygnowałem z nabycia tej karty, jednak projekt na tyle mnie zainteresował, że postanowiłem dokładniej przyjrzeć się pracy tego zdolnego Słowaka i comiesięcznie wspierać jego kolejny projekt finansowo. Postępy prac śledziłem dzięki stronie *http://www.amigaportal.cz*.

Oszołomiony powodzeniem pierwszego projektu „Boboo" podjął się stworzenia kolejnego akceleratora opartego na procesorze 68020. Dodatkowego smaczku dodawał fakt, że produkt Borisa wydawał się, jak dla mnie, doskonałą alternatywą dla turbin ACA spod szyldu Individual Computers. Boris początkowo zakładał ukończenie Furii EC020 w pierwszym kwartale 2013 roku, ale był niezadowolony z oprogramowania sterującego akceleratorem określając je jako złe i niewystarczająco dobre. Osiągi pozostawiały wówczas wiele do życzenia – nieco ponad 5 MIPS, ale warto było czekać. 9 grudnia 2013 roku otrzymałem potwierdzenie zamówienia. Po wpłaceniu kwoty 90 euro plus koszty wysyłki, karta dotarła do mnie kilka dni przed świętami i była doskonałym prezentem – tym bardziej, że otrzymałem podziękowanie widoczne na PCB.

Furia EC020 28 MHz posiada 9,5 MB Fast pamięci DRAM (z możliwością dodania 1,5 MB) o czasie dostępu 60 ns, dzięki czemu karta pracuje szybko osiągając wynik 5,89 MIPS. Z kartą CF 512 MB Kingston Elite Pro osiąga na standardowym scsi.device z FFS około 1,7 MB/s. Jako dodatkowe oprogramowanie używam także BlazeWCP, FText oraz FBlit. Nie bez znaczenia pozostaje także FPU pracujący z tą samą częstotliwością, co procesor. Miedziane radiatory widoczne na pamięciach zamontowałem dodatkowo, gdyż fabrycznie takich nie ma, a po kilku godzinach pracy robi się bez nich naprawdę gorąco. Do używania pakietu WHDLoad zaleca się wersję od 17.x wzwyż, gdyż z innymi mogą występować problemy – zwłaszcza dotyczy to starszych gier, gdzie zachodzi czasami konieczność wyłączenia FPU lub cache.

Boris nie zapomniał także o oprogramowaniu o nazwie FuriaTune autorstwa Martina Kuchinki będącym nawiązaniem do ACATune. Kod źródłowy programu został napisany w assemblerze. Obecnie można używać wersji 1.2. Sposób użycia programu jest banalny – wystarczy skopiować go do katalogu C: partycji systemo-

Testy pamięci wykonany w programie AIBB. Jak widać, konkurencja daleko w tyle.



obecnie dodatkowego sprzętu, jak np. A603 lub A604, aby móc wypróbować współpracę z IndivisionECS oraz montowanych na clockporcie dodatków Silversurfer, Delfina tudzież Catweasel MK2. Mocno ubolewam też nad brakiem 030 „na pokładzie" pozwalającym na rozbudowę o SubwayUSB. Nie bez znaczenia pozostaje też obecność koprocesora matematycznego, który znajdzie zastosowanie w niektórych demach, programach graficznych, grach doomopodobnych, symulatorach.

wej i wywołać z CLI z odpowiednim parametrem. Poniższe opracowanie wykonałem na podstawie komend zawartych w FuriaTune 1.1.

- **?** – wyświetla listę komend,
- **status** – wyświetla aktualną konfigurację Furii,
- **reboot** – restart Amigi,
- **default** – przywraca ustawienia fabryczne (czasami może być wymagany restart),
- **fpu on/off** – w zależności od opcji włącza lub wyłącza FPU,
- **cpu cache on/off** – w zależności od opcji włącza lub wyłącza cache procesora,
- **shadowrom** – włącza dostęp do 32-bitowej pamięci,
- **boardrom** – wyłącza dostęp do 32-bitowej pamięci,
- **maprom** *file* – wczytuje ROM z pliku (*file* to nazwa pliku),
- **addmem** – dodaje 1,5 MB pamięci.

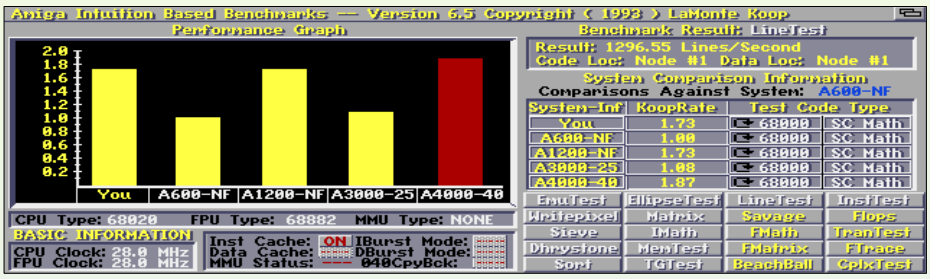
Dodatkowo przytrzymanie przez 3 sekundy klawiszy CTRL+LAmiga+RAmiga włącza tryb PCMCIA friendly, czyli uzyskujemy 5,5 MB Fast.

Ciekawe efekty daje użycie kombinacji komend **shadowrom** i **addmem** – ilość MIPS wzrasta do 5,96, zyskujemy 1,5 MB pamięci oraz wzrasta transfer do ponad 1,9 MB/s. Dodatkowo próbowałem aktywować funkcję **ide-on**, ale efekt był żaden – być może zostanie to kiedyś dopracowane, gdyż widziałem obrazki z Sysinfo, gdzie dzięki tej komendzie transfer wynosił ponad 4 MB/s!

Karta jest świetną propozycją dla każdego, kto trochę poważniej myśli o korzystaniu z A600. Za całkiem rozsądną kwotę nabywamy wysokiej jakości produkt stanowiący na chwilę obecną czołówkę kart turbo dla sześćsetki. Tak wyposażona Amiga pozwala komfortowo pracować w środowisku Workbencha – kopiowanie plików przebiega szybko, praca z programami, takimi jak Deluxe Paint, DOpus, Filemaster, Scala należy do przyjemności. Zgodność z pakietem WHDLoad jest bardzo wysoka i nie zauważyłem żadnej różnicy w zgodności z grami względem innych kart, które dane mi było używać z Amigą 1200 (Blizzard IV, M-tec, Elbox 1230). Niestety nie posiadam

P.S. Obecnie trwają prace nad Furia EC020 Rev2. Ma to być ten sam typ karty tylko taktowany 33 MHz oraz 40 MHz uzyskując osiągi na poziomie odpowiednio 7 MIPS oraz 8.41 MIPS. Wygląda to bardzo obiecująco.

Mirosław Arciszewski

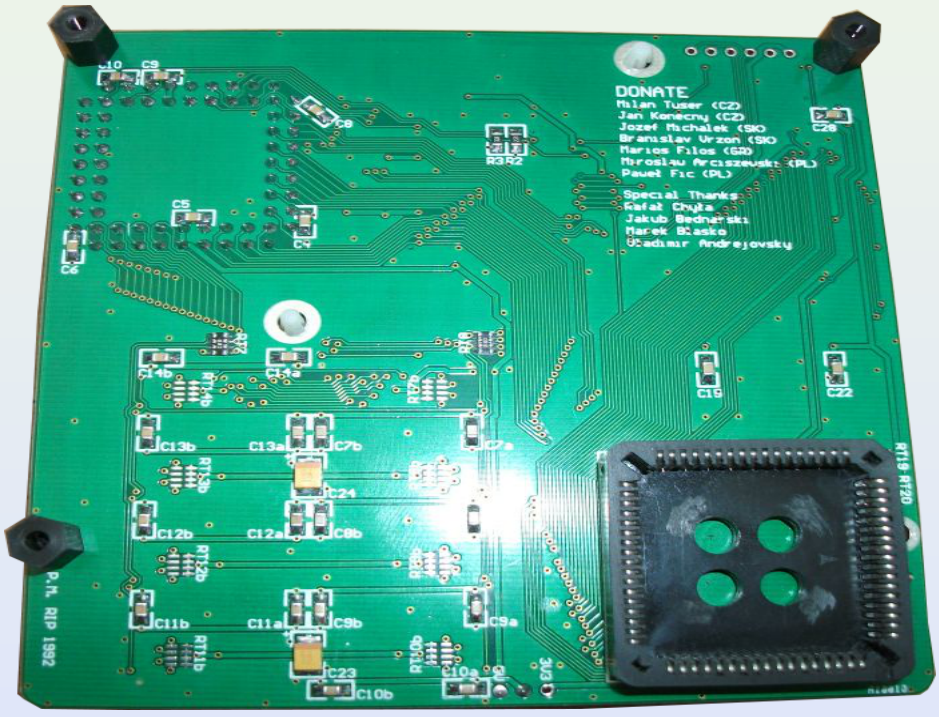


Oficjalna strona projektu::

http://www.kuchinka.cz/furia/

- szybkość działania (nieznacznie tylko ustępuje ACA030),
- szybki dostęp do pamięci (ponad 14 MB/s),
- FPU,
- cena, cena, cena,
- wysoka jakość wykonania,
- bardzo dobre wyniki transferów.

- brak pisemnej gwarancji oraz jakiegoś fajnego opakowania,
- „tylko" 11 MB dodatkowej pamięci.



Druga strona karty. Jak widać, wytłoczone zostały na niej nazwiska osób, które przyczyniły się do finansowego wsparcia projektu.

BetterWB czy ClassicWB?

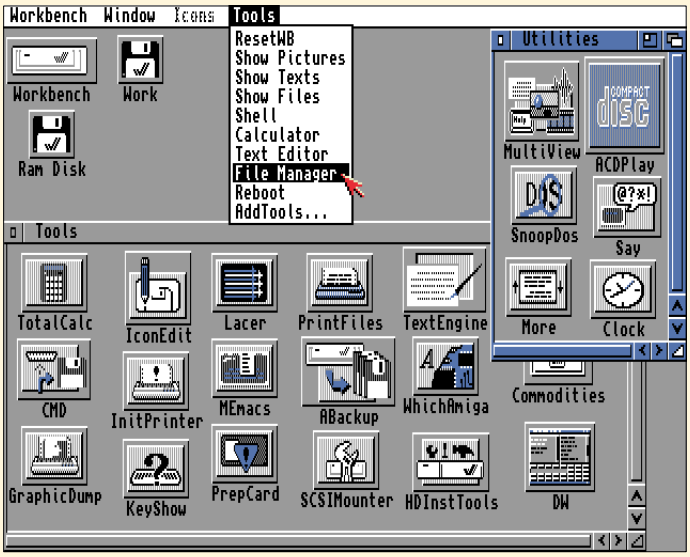
Dawno już minęły czasy, kiedy użytkownik Amigi klasycznej mógł się zadowolić gołym systemem AmigaOS, instalowanym ze sfatygowanych dyskietek. Nawet dwie ostatnie wersje systemu dla klasycznej Amigi – 3.5 i 3.9 kuleją już w wielu miejscach, takich jak obsługa dużych dysków twardych, brak wygodnego filemanagera czy stare wersje bibliotek. W związku z oczywistym brakiem szans na rozwój systemu AmigaOS na procesory 68k, w sukurs przyszli jak zwykle sami amigowcy, tworząc gotowe pakiety oprogramowania i łatek, przygotowane do instalacji na „goły" system. Nie bez znaczenia pozostaje też fakt stale rosnącej popularności emulatorów Amigi klasycznej, których użytkownicy zamiast instalować i konfigurować system od zera, wolą korzystać z gotowych rozwiązań. W niniejszym artykule pragnę opisać dwa popularne pakiety wzbogacające AmigaOS, czyli *BetterWB* i *ClassicWB*. Oba te rozwiązania łączy fakt, że można je z powodzeniem wykorzystać nawet na niezbyt rozbudowanych konfiguracjach sprzętowych, a różni natomiast sposób podejścia do kwestii rozbudowy systemu i ilości dostępnych po zainstalowaniu programów.

BetterWB

Na pierwszy ogień pójdzie prostszy z pakietów, czyli BetterWB. Pakiet ten został pomyślany, jako rozszerzenie dla AmigaOS3.0 i 3.1, z przeznaczeniem do uruchamiania na prawdziwym sprzęcie, z założeniem, że wystarczy procesor 68000, standardowa ilość pamięci RAM oraz 10 MB miejsca na twardym dysku. Nie znajdziemy tu więc wizualnych wodotrysków, tony programów na każdą okazję czy pakietu gier. Znajdziemy za to masę usprawnień, wspomagających codzienne użytkowanie lawendowej Amigi.

Instalacja

W archiwum znajduje się sześć obrazów dysków, w formacie DMS bądź ADF, nazwanych *„Misc3.1"*, *„Tools3.1"*, *„Bonus3.1"*, *„Docs31"*, *„Icons3.1"* oraz *„Drivers3.1"*. Instalacja powinna odbywać się na „gołym" systemie i zaczyna się od dyskietki *„Misc3.1"*. Proces instalacji, w zależności od posiadanego w Amidze procesora, może potrwać do kilkunastu minut jako, że na dyskietkach znajdują się archiwa LhA, które skrypt instalacyjny musi najpierw rozpakować. Po instalacji Amigę wystarczy zresetować i system jest gotowy do użytku.

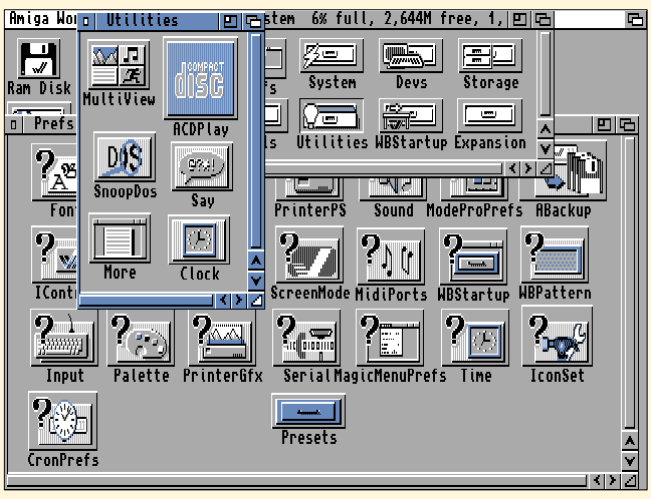


Użytkowanie

Na pierwszy rzut oka w systemie nie widać wielu zmian, na ekranie pozostają standardowe szarości, jednak już po kliknięciu na ikonkę dysku systemowego naszym oczom ukazują się poprawione (aczkolwiek nadal 4-kolorowe) ikonki. Większość zmian, jakie zaszły w systemie, widać po otwarciu kolejnych katalogów systemowych czy zajrzeniu do startup-sequence. Najważniejsze dodane programy to, np. *HDInstTools*, czyli program do przygotowywania i partycjonowania twardych dysków, będący wygodnym i bezpiecznym zamiennikiem systemowego HDToolboxa. W katalogu Tools znajdziemy także lekki manager plików *DirWork*, edytor tekstu *TextEngine*, kalkulator *TotalCalc* czy też program diagnostyczny *WhichAmiga*. W katalogu Utilities umieszczony został odtwarzacz płyt CD oraz znany monitor aktywności systemu *SnoopDos*. BetterWB instaluje także między innymi oprogramowanie do obsługi plików ADF *tsGui*, rozszerzony wiersz poleceń *Zshell* i kopiera dysków *SuperDuper*.

Poza zmianami widocznymi dla oka i naszego kursora myszki, wiele usprawnień zachodzi też w samym systemie. Wszystkich nie sposób tutaj wymienić, jednak do najważniejszych należą:

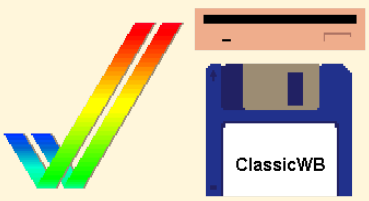
- nowe datatypy (gif, jpg, bmp, png, pcx, ilbm, tga, tiff, wav, cdxl),
- wiele dodanych commodities, takich jak *Magic Menu* czy *Win2Front*,
- paker *XPKMaster*,
- najnowsze wersje archiwizerów LHA, LZX, ZIP, DMS i innych,
- FastFileSystem w wersji 44.5,
- pakiet Fat95 do obsługi dysków w formatach FAT i FAT32,
- wiele zaktualizowanych bibliotek i poleceń systemowych (np. Setpatch icon.library i inne),
- zainstalowany i gotowy do wykorzystania pakiet filesystemu PFS3,
- dodatkowe sterowniki drukarek,
- możliwość szybkiej podmiany ikon na kolorowe GlowIcons,
- i wiele innych...



BetterWB

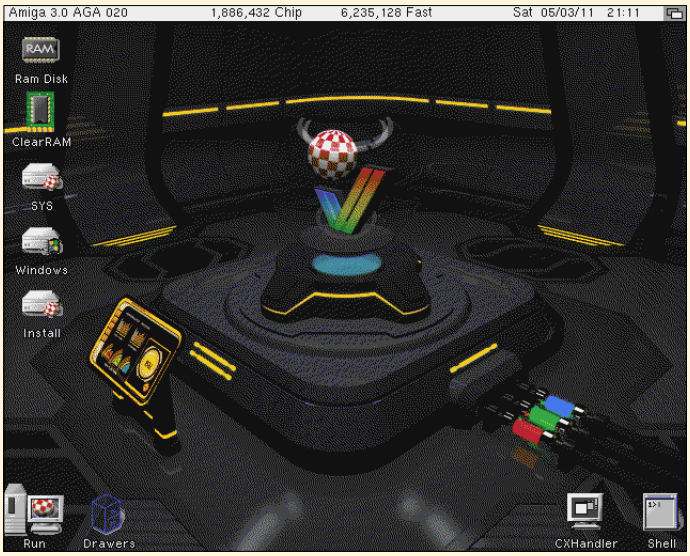
Czy warto?

Pakiet BetterWB na pewno jest warty uwagi użytkowników posiadających mało rozbudowane Amigi. Osobiście wykorzystałem go stawiając system koledze na A1200 z zaledwie 2 MB pamięci Fast na wolniutkiej karcie PCMCIA. BetterWB przy takiej konfiguracji staje się bardzo funkcjonalnym rozszerzeniem systemu, a dzięki instalacji pakietu zyskujemy też sporo czasu, który stracilibyśmy na pobieranie z Aminetu potrzebnych programów i łatek oraz ręczne ich instalowanie. Narzędzia, takie jak *HDInstTools* czy manager plików, są po prostu od razu pod ręką, a jednocześnie system nie pożera dużo pamięci Ram i nadal zajmuje mało miejsca na dysku. Nie bez znaczenia jest też fakt, że pakiet BetterWB jest bardzo często aktualizowany – w ciągu pierwszego półrocza 2014 roku były już cztery aktualizacje.



ClassicWB jest zupełnie innym pakietem niż opisany powyżej BetterWB. Oczywiście idea ulepszenia systemu operacyjnego jest niezmienna, za to twórcom pakietu przyświecał zupełnie inny zamysł. Już samo wejście na stronę projektu może przyprawić o lekki zawrót głowy, bo do wyboru mamy na dzień dobry aż osiem (!!!) wersji pakietu:

- Lite – wersja przeznaczona do używania w



Wersja Full

trybach 8-16 kolorów w rozdzielczości 640x256 dla docelowego systemu, jakim jest goła A1200,
- Full – rozbudowana wersja powyższego pakietu, przeznaczona dla A1200 z 4 MB pamięci Fast,
- ADV – modyfikacja wersji Full, przeznaczona dla rozdzielczości 640x512 w minimum 16 kolorach,
- ADVSP – wersja dla dopalonych A1200,
- P96 – wersja dla kart graficznych w trybach 16 i 32-bit,
- UAE – gotowy system dla emulatorów,
- OS3x – wersja dla systemów AmigaOS 3.5 i 3.9,
- 68k – najprostsza wersja pakietu, która zadowoli się procesorem 68000, 2 MB pamięci i kickstartem 2.0 (wersja GAAE).

Jak widać z powyższej listy, wybór wersji jest szeroki i powinien zadowolić każdego: od użytkownika A500 z ACA500, przez użytkowników A600 z rozszerzoną pamięcią i standardowym kickstartem, po highendowe Amigi klasyczne i użytkowników UAE.

Instalacja

I tutaj są pierwsze schody. Mimo, że większość wersji pakietu przeznaczona jest dla „prawdziwej" Amigi, to instalację najłatwiej przeprowadzić pod emulatorem. Dlaczego? Otóż w archiwum z ClassicWB znajduje się plik w formacje *.hdf, czyli jest to plikopartycja emulatora UAE. Według autorów pakietu, ma to na celu ułatwienie instalacji i łatwość dostosowania pakietu do swoich potrzeb przed transferem na prawdziwą Amigę. Nie do końca zgadzam się z tą argumentacją, ponieważ jako zatwardziały amigowiec nie lubię do amigowania używać peceta… ale co zrobić? Takie czasy, panowie... Tak czy inaczej, w celu zainstalowania pakietu, niezbędne jest wykorzystanie emulatora oraz posiadanie odpowiednich plików z kickstartem (w zależności od instalowanej wersji ClassicWB wymagany kickstart do 2.0 do 3.1). System instaluje się dość łatwo, bootując emulator z plikopartycji i w odpowiednim momencie umieszczając w „stacji" plik ADF z obrazem dysku z odpowiednią wersją AmigaOS. Osobną kwestię stanowi przetransferowanie plików z systemem na prawdziwą Amigę. Ja osobiście użyłem karty CF, podłączonej do peceta, którą sformatowałem pod emulatorem jako dysk amigowy. Innym wyjściem może być podłączenie amigowego dysku bezpośrednio do peceta czy transfer plików na płycie CD.

Użytkowanie

W moim opisie skupię się na wersji ClassicWB, której używam na Amidze 600, czyli GAAE. Jest to bodajże jedyna wersja pakietu, która zadowoli się kickstartem 2.x. Nie zawiera może ona wszystkich wizualnych wodotrysków, obecnych w wersjach na lepsze konfiguracje, jednak podstawowe zasady działania pakietu są podobne dla każdej jego edycji. Po odpaleniu nowo zainstalowanego systemu od razu widzimy wiele usprawnień w stosunku do „gołego" Workbencha. Domyślnie uruchomione programy i łatki to:

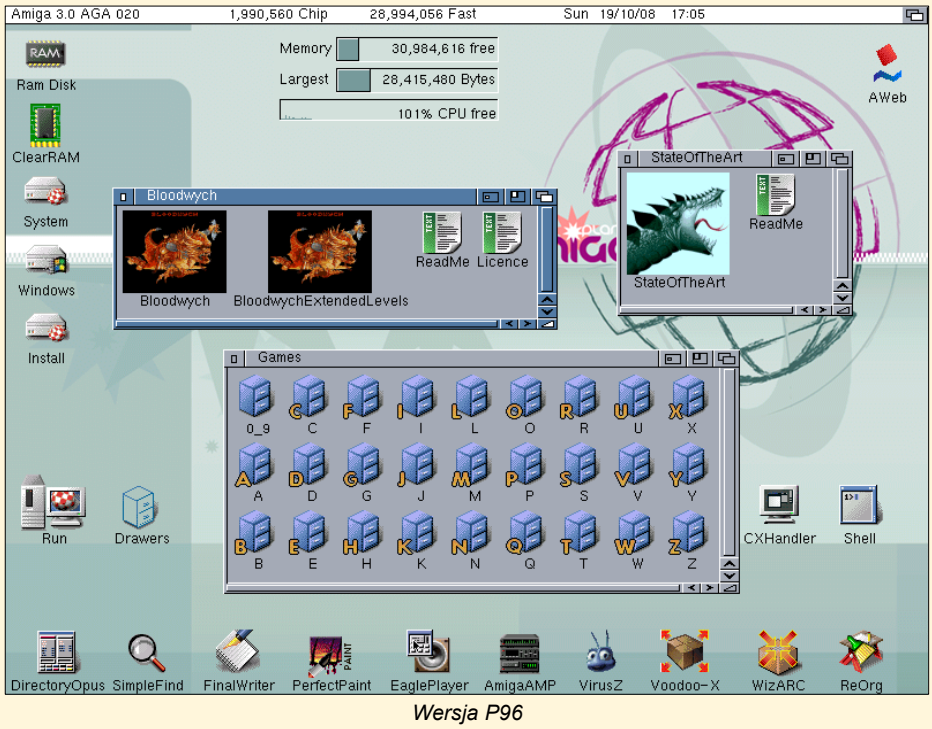
- *ToolsDaemon* – dodający multum nowych komend do menu w górnej belce Workbencha. W domyślnej konfiguracji ClassicWB mamy tam szybki dostęp do ustawień systemowych, narzędzi do archiwizacji plików, edycji tekstu, otwierania CLI, a także szybkiego uruchamiania wielu domyślnie zainstalowanych programów (playery, przeglądarki obrazków, *DOpus4*, *SnoopDos* i wiele innych);
- *AssignWedge* – łatka umożliwiająca szybkie przypisywanie dysków logicznych, w przypadku ich wywołania przez programy, eliminuje potrzebę używania komendy *assign;*
- *MagicMenu* – powszechnie używany przez amigowców program, dzięki któremu do opcji dostępnych w pull-down menu, na górnej belce Workbencha, mamy dostęp w każdym miejscu ekranu;
- *AmiDock* – umieszczony domyślnie u dołu ekranu pasek szybkiego uruchamiania programów;
- dodanie na blat Workbencha ikonek *Shell*, *CX Handler* (obsługa commodities), *Clear Ram* (czyszczenie pamięci ze „śmieci"), *Run* (uruchomienie *ButtonMenu* z szybkim dostępem do programów);
- *MagicWB* – pakiet 8 kolorowych ikon, w wersji przystosowanej do rozdzielczości 640x256. Włączona jest także obsługa *NewIcons;*
- Pakiet MUI;
- *SmartWin* – łatka przyspieszająca otwieranie okienek.

Oczywiście zmian i usprawnień jest dużo więcej, wszystko dokładnie opisane jest w dokumentacji pakietu, więc nie ma sensu wymieniać tutaj całości. Poza usprawnieniami czysto systemowymi, ClassicWB oferuje także spory pakiet oprogramowania. Wystarczy powiedzieć, że po instalacji wszystko zajmuje ok. 45 MB miejsca na dysku. Przeglądając katalogi systemowe trafimy chociażby na następujące programy:

- katalog System: *AmiDock*, *ButtonMenu*, *AGLaunch*, *SmartCache*.
- katalog Utilities: *SnoopDos*.
- katalog Storage: dodatkowe sterowniki drukarek, monitorów, mapy klawiatury itp.
- katalog Help: pełna dokumentacja wszystkich zainstalowanych dodatków.
- katalog Tools: *TTManager*, *IconImage*.

Na partycji systemowej odnajdziemy także nowe katalogi z zainstalowanym oprogramowaniem:

- Programs: *DirOpus4, DiskSalv, DpaintIV, FMSsys, SuperDuper, HippoPlayer, OctaMED, PeelIcons, SysInfo, Trans Write, Georg, TSGui, VirusZ*. Dodatkowo w katalogu Programs odnajdziemy pakiet gier Public Domain.
- Icons – zestawy ikon na każdą okazję.
- My Files – zestaw oprogramowania do obsługi dużych dysków twardych (różne wersje *scsi.device*, *FFS_v43.20, PFS3, HDInstTools* i inne), zestaw modułów w katalogu Music, narzędzie do dzielenia i łączenia plików (*Splitz* i *Joinz*).



Wersja P96

Dużą zaletą ClassicWB jest także domyślne skonfigurowanie pod użytkowanie WHDLOAD. Aby wszystko działało, wystarczy zapisanie w katalogu devs:kickstarts odpowiednich plików z kickstartami, niezbędnych do uruchamiania gier i dem.

Wszystkie powyższe usprawnienia mają swoją jedną wadę: zajmują dużo pamięci RAM, co przy konfiguracjach z mniejszą ilością dostępnej pamięci może być problematyczne. Autorzy pakietu przewidzieli jednak rozwiązanie tego problemu. Otóż, jeśli podczas bootowania Amigi przytrzymamy wciśnięty prawy klawisz myszy, trafimy do ClassicWB BootMenu, gdzie będziemy mogli wybrać parę alternatywnych opcji uruchomienia Amigi, takich jak: CLI, Minimum Startup, Safe Startup, Virus scan i inne. Oczywiście menu jest w pełni konfigurowalne i nic nie stoi na przeszkodzie, aby dodać do niego nowe opcje (np. ja dodałem opcję uruchomienia odpalarki WHDLOAD – *X-Bench*).

Podsumowanie

Jak widać, ClassicWB jest pakietem o bardzo dużych możliwościach, w znaczny sposób usprawniającym i modyfikującym pracę z systemem Amigi. Dla niektórych wadą może być sposób jego instalacji, wymuszający w praktyce użycie emulatora, jednak nie jest to kwestia dyskwalifikująca ClassicWB. Pakiet jest bardzo rozbudowany i w powyższym artykule nie zamieściłem na pewno informacji o wszystkich zmianach, jakie wprowadza. Na pewno do jego największych zalet należą ogromne możliwości konfiguracji, duży wybór oprogramowania dostępnego „pod ręką” czy, dzięki dużemu zróżnicowaniu dostępnych wersji pakietu, możliwość instalacji na praktycznie każdej konfiguracji Amigi. Pakiet mogę polecić każdemu amigowcowi, który niekoniecznie chce się bawić w „ręczne” konfigurowanie systemu oraz wyszukiwanie i instalację łatek i programów, które ułatwiają nasze amigowanie.

Mateusz „Twardy” Eckert

ClassicWB UAE

ClassicWB GAGE

BetterWB

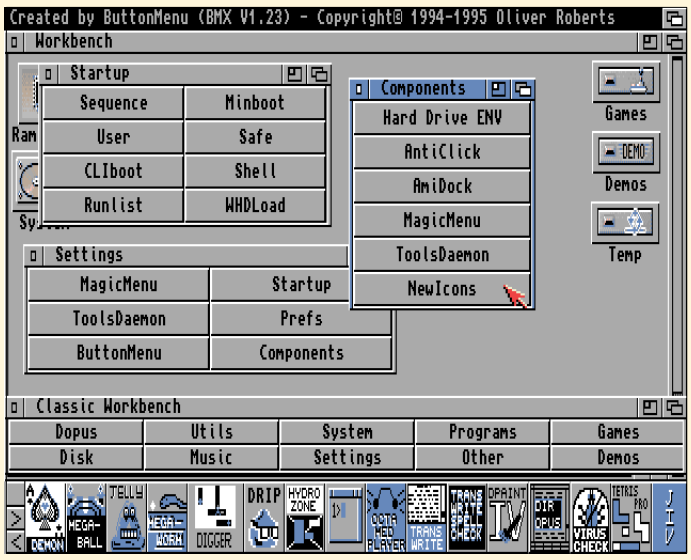
Autorem pakietu BetterWB jest niejaki Gulliver, który rozwija swój projekt z czystej chęci stworzenia czegoś dla amigowej społeczności.

http://lilliput.amiga-projects.net/BetterWB.htm

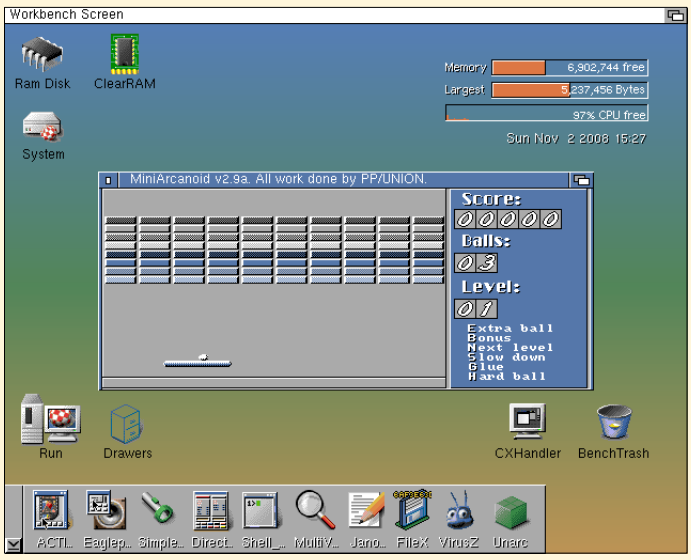
ClassicWB

Projekt, za który odpowiedzialny jest Bloodwych – członek społeczności portalu English Amiga Board. Pakiet można pobrać ze specjalnie wydzielonej podstrony na serwerze, a dyskusję na jego temat prowadzić na forum będącym częścią wspomnianego portalu. Wszelkie sugestie, uwagi, błędy można zgłaszać właśnie tam, podobnie jak i tam w pierwszej kolejności autor zamieszcza informację o nowych wersjach.

http://classicwb.abime.net/
http://eab.abime.net/forumdisplay.php?f=61



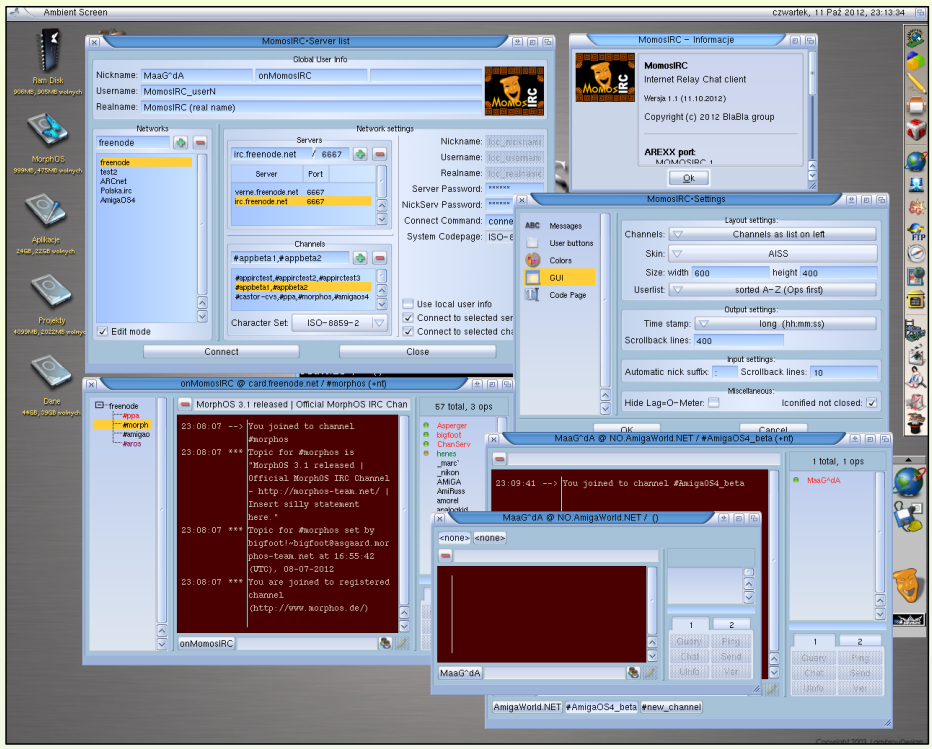
ClassicWB GAGE



ClassicWB OS3.9

MomosIRC

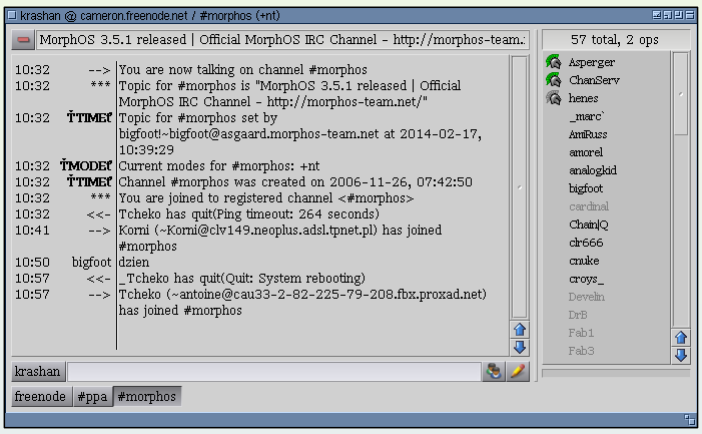
Tradycyjnie przyjęło się uważać, że do czego, jak do czego, ale do pogaduszek na IRC-u systemy amigowe oprogramowane są znakomicie. Bliższe przyjrzenie się sytuacji ujawnia jednak, że ta opinia bazowała na jednym programie, mianowicie *AmIRC-u*. Ten zaś, choć niewątpliwie dobry, a od jakiegoś czasu również darmowy, zaczął zdradzać objawy wieku. Podstawowy problem, szczególnie istotny dla użytkowników spoza zachodnioeuropejskiej strefy językowej, to problem kodowania znaków. Na większości kanałów polskojęzycznych standardem stało się kodowanie UTF-8. Częściowym rozwiązaniem problemu Unikodu dla użytkowników MorphOS-a stała się wtyczka do *AmIRC-a* wydana przez Thomasa Bodliena. Choć zasadniczo spełnia swoje zadanie, ma kilka wad – na przykład wywołuje problemy z kolorowaniem wypowiedzi. Próba dodania obsługi kodowań do samego *AmIRC-a* też nie jest prosta, ze względu na dość nieuporządkowany projekt programu.



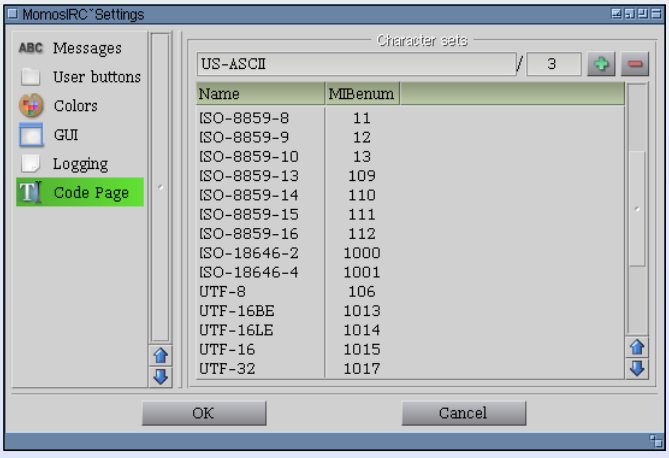
Na górze kilka okienek programu otwartych jednocześnie.

Nic dziwnego, że w tej sytuacji kilku programistów próbowało swoich sił chcąc napisać alternatywny program do IRC. Część z tych prób zakończyła się mniej lub bardziej działającymi produktami: *WookieChat*, *Sermonatrix*, mamy też port konsolowego *IRSSI*. Mamy też wreszcie nasz polski produkt, tytułowego *MomosIRC-a*, napisany przez Mariana Guca. Autor początkowo miał zamiar przeportować napisany na AROS-a program *AIRCOS*, ale najwyraźniej zachwycony jakością jego kodu, zdecydował się na napisanie własnego programu.

MomosIRC jest w pełni funkcjonalnym programem, co bardzo ważne nie mającym żadnych problemów z obsługą wielu kodowań znaków, z unikodem włącznie. Co więcej, na różnych kanałach IRC czy oknach prywatnych możemy w tym samym czasie pracować z różnymi kodowaniami, możemy je również dynamicznie zmieniać wewnętrzną komendą, gdy na przykład zorientujemy się, że rozmówca pisze w ISO-2 czy nawet w AmigaPL. Program korzysta z systemowej biblioteki *charset library* i obsługuje wszystkie znane jej kodowania. Trzeba jednak pamiętać o tym, że póki co MorphOS nie ma obsługi rysowania tekstu z pełnym zestawem znaków Unicode, chociaż autor zapowiada ominięcie tego problemu przez użycie biblioteki *TTEngine*. Niemniej problem pisania z polskimi znakami na kanałach IRC używających kodowania UTF-8 jest w pełni rozwiązany.



Główne okno programu.



Okno ustawień programu.

Do rozwiązania pozostaje natomiast wiele drobnych niedogodności, z jakimi spotka się użytkownik *MomosIRC-a*. Nie dyskwalifikują one rzecz jasna programu, tym bardziej, że autor wciąż go ulepsza. Jednak osoba przyzwyczajona do wielu udogodnień z programu AmIRC, może być nieco rozczarowana. Jednym ze źródeł problemów jest użyta do wyświetlania rozmowy na kanale zewnętrzna klasa MUI NList. Autor ma zamiar ją wyeliminować, ale póki co przyczynia się ona do pewnej niestabilności programu. Od czasu do czasu *MomosIRC* lubi się zawiesić, podczas gdy *AmIRC,* na tej samej maszynie, jest stabilny jak skała. NList ma również problemy z kopiowaniem tekstu, zaznaczanie jest problematyczne, a czasem kopiuje się nie to, co zaznaczyliśmy. Nie ma też funkcji automatycznego wyróżniania adresów URL oraz automatycznego otwierania ich w przeglądarce internetowej. Pomniejszym problemem jest brak obsługi kolorowania tekstu. Nie działa też obsługi przesyłania plików za pomocą DCC, chociaż ta funkcja często i tak nie działa w dobie firewalli i stosowania lokalnych adresów IP.

Słodko więc nie jest, ale nie przeszkadza mi to używać *MomosIRC-a* na co dzień jako podstawowego programu do czatowania. Tym bardziej, że w tej chwili program Mariana Guca jest chyba jedynym aktywnie rozwijanym klientem IRC dla MorphOS-a, więc w przyszłości może być tylko lepiej :-).

Grzegorz Kraszewski

na gorąco

Na przełomie maja i czerwca ukazała się kolejna wersja pakietu AmiKit – prekonfigurowanej, darmowej dystrybucji środowiska AmigaOS – oznaczona już numerkiem 7. Postanowiłem się jej nieco bliżej przyjrzeć, zwłaszcza, że mogłem przetestować ją na specjalnej, wówczas jeszcze niedostępnej dla zwykłego użytkownika, wersji *WinUAE 2.8.1 AmiKit preview*. Na wstępie należy odnotować, że początkowo słowo „darmowa" było nieco na wyrost – nie było możliwe pobranie pakietu zwyczajowymi metodami, gdyż autor, Jan Zahurancik, zdecydował o przygotowaniu limitowanej edycji dystrybuowanej na specjalnie przygotowanym „paluchu USB". Można było ją nabyć w kilku wariantach cenowych, w zależności od wysokości dotacji, jaką chcieliśmy przekazać autorowi za wkładany wysiłek w rozwijanie pakietu. Po dwóch tygodniach od tego momentu wydana została również wersja do pobrania w tradycyjny sposób w wersjach przeznaczonych dla systemów Windows, Linux, Mac OS, Android, a także nieco okrojona wizualnie dla Amig klasycznych. Jak już wspomniałem, na potrzeby tego artykułu skupię się na tej pierwszej, od której wszystko się zaczęło. Przejdę przez proces instalacji pakietu, a następnie podzielę się moimi odczuciami.

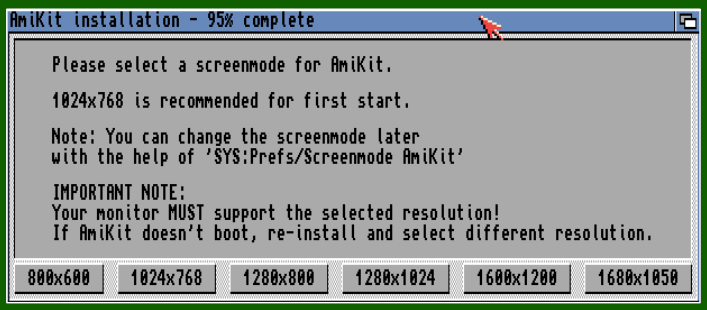
Instalacja

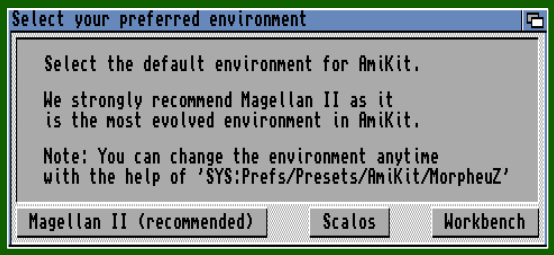
Pakiet „zassałem" ze strony domowej projektu i uzbrojony w laptopa, przystąpiłem do instalacji. Sam proces jest prosty i intuicyjny – jesteśmy praktycznie prowadzeni za rączkę krok po kroku. Aby przystąpić do zabawy, musimy zapewnić sobie posiadanie pod ręką amigowego kickstartu (ROM) 3.1 zgranego do pliku i mieć w pogotowiu płytę CD/DVD z system amigowym AmigaOS 3.5 lub AmigaOS 3.9 (czy choćby ich odmianę AmigaOS XL lub pakiet Amiga Forever). Dyskietki z systemem 3.x nie pozwalają na instalację pakietu. Amigowy skrypt instalacyjny jest zlokalizowany i dostępny jest do wyboru język polski.

Plik kickstartu 3.1 najlepiej skopiować do danego katalogu w szufladzie Amikita. „Będąc szczęśliwym posiadaczem" pakietu Amiga Forever 2012 Plus w postaci obrazu ISO kupionego na promocji, natychmiast się go dogrzebałem w czeluściach przenośnego dysku twardego. Przed uruchomieniem windowsowego instalatora Amikita zainstalowałem AF 2012 Plus w wirtualnym napędzie, zgodnie z zaleceniami skryptu instalacyjnego.

Po znalezieniu amigowego ROM-u i wskazaniu systemu na CD/DVD, instalator pakietu Amikit ochoczo przystąpił do kopiowania koniecznych plików na twardy dysk.

Sam proces kopiowania amigowego systemu z CD/DVD zajmuje nieco czasu (do przeniesienia jest ponad 35000 plików), o czym instalator grzecznie przypomina. Następny etap to konfiguracja WinUAE, rozdzielczości amigowego blatu (zalecana rozdzielczość 1024x768) i środowiska do wyboru spośród Magellana II, Scalosa i Workbencha. Tutaj autor również rekomenduje wybór – Magellan II (w trybie zamiennika Workbencha). Zdecydowałem się na tę opcję.





Kilka chwil później zobaczyłem ekran bootujący Amikitowej Amigi w wysokiej rozdzielczości wraz z paskiem postępu, a następnie moim oczom ukazał się blat Amikita w pełnej okazałości. Za pomocą amigowego programu *ScreenMode* ustawiłem rozdzielczość ekranu do bardziej praktycznej w formacie 16x9, pasującej do mojego notebooka, po czym przystąpiłem do oględzin Amikitowej Amigi.

Wrażenia

Jako, że piszę te słowa jeszcze niemalże na gorąco, tuż po zainstalowaniu pakietu, moje pierwsze odczucia są ambiwalentne, że tak to ujmę. Graficznie czy też wizualnie cały desktop prezentuje się bardzo ładnie. Co więcej – widać, że autor musiał rzeczywiście poświęcić mnóstwo czasu na selekcję i instalację amigowego oprogramowania. Do tego, co od razu rzuca się w oczy, wszystko jest ładnie dopieszczone i wstępnie skonfigurowane, aby zaoszczędzić początkującemu użytkownikowi tej żmudnej konieczności, a i nawet uciążliwości.

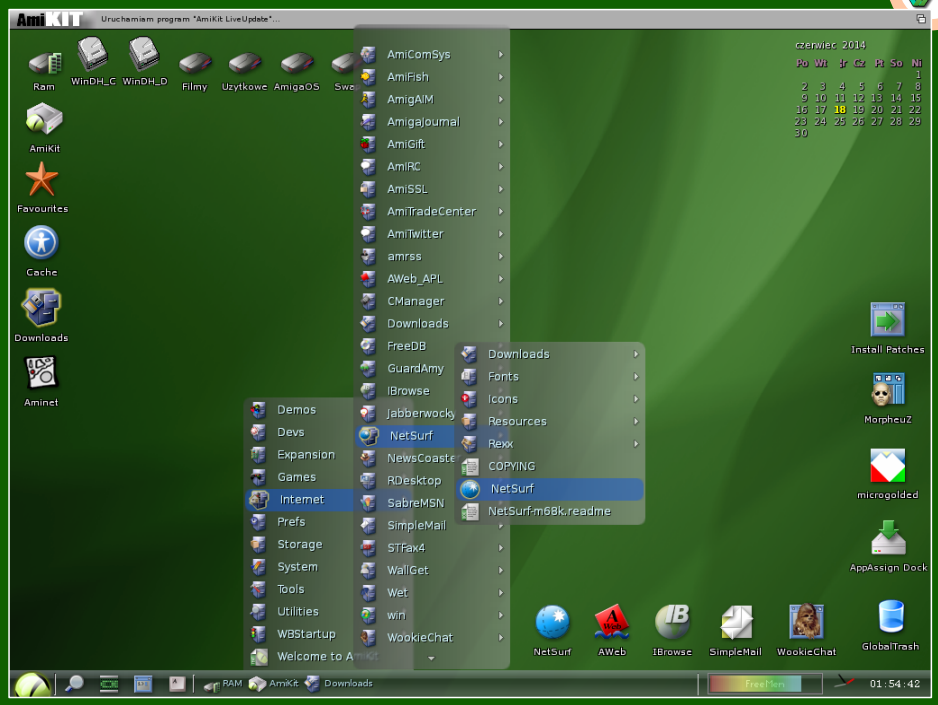
Ale, niestety, pojawiły się i pierwsze „schodki". Jako, że „apetyt rośnie w miarę jedzenia", a wcześniej ustawiłem środowisko na Magellana II jako zamiennik Workbencha, oczekiwałem na odlotową konfigurację. I tu się w dużej mierze zawiodłem. Co mi się na początku rzuciło w oczy, to zbyt małe czcionki ustawione w listerach. W tym przypadku wystarczyły sekundy „grzebania" w ustawieniach Magellana, aby rzecz poprawić, jednak trudniejsza sprawa związana jest z ikonkami na belce listera, które są zdecydowanie za małe w stosunku do zalecanej rozdzielczości blatu, nie wspominając już jak to musi wyglądać na wyższych rozdzielczościach. Przydałby się również, oprócz większego rozmiaru tych ikonek, także lepszy efekt 3D wciśnięcia ikonki, a nie tylko jej zacienienie. Według mnie, autor umieścił także zbyt dużo narzędzi w belce narzędziowej listera i dlatego są one takie małe, aby jak najwięcej ich tam pomieścić. Kwestia dyskusyjna też samego doboru ikonek czy np. ikonka z symbolem „i" najbardziej się nadaje do zmiany trybu wyświetlania listera. Podobnie problematyczne jest ustawienie czcionek w

AmiKit w trakcie uruchamiania

menu belki tytułowej Magellana czy rozwijanym pop-up menu po wciśnięciu prawego przycisku myszy na blacie. Czcionki tam wykorzystane są zdecydowanie zbyt małe i przez to małe czytelne, przynajmniej na ekranie o rozmiarze 15 cali.

Z innych bardziej praktycznych rzeczy brakuje mi, na przykład powrotu do dostępnej listy napędów z poziomu listera. W tej chwili po dwukliku lewym przyciskiem myszy na blacie, pojawia się lister z napędami. Po wyborze któregoś z tych napędów, nie można się później cofnąć w tym samym listerze do listy napędów. Jako, że działamy na emulatorze przydałoby się jeszcze tworzenie pliku HDF z danego katalogu. Jest funkcja tworząca obraz ISO, ale niestety nie działa. Rzuca się też w oczy brak konfiguracji magellanowego dwukliku prawego przycisku myszy na blacie. Autor wykorzystał zamiennie program *MagicMenu* i niestety, zdublował w tym menu ustawienia z belki tytułowej Magellana, a więc rzeczy której raczej rzadziej się wykorzystuje w codziennej praktyce. Brakuje mi też lepszej konfiguracji wydarzeń systemowych śledzonych przez Magellana (Scripts), gdyż tylko dwa wydarzenia mają przypisane im funkcje. Kontrowersyjne jest – moim zdaniem – wykorzystanie programu *AmiStart* w miejsce magellanowego menu Start. Krótko mówiąc, jeśli Magellan ma zastępować nam Workbencha, to część wykorzystanych w pakiecie programów jest raczej zbędna.

Oczywiście wiele rzeczy z założonej w Amikicie konfiguracji amigowego desktopu można samemu poprawić czy zmienić, ale do tego potrzebna jest raczej nieco głębsza niż tylko powierzchowna znajomość systemu amigowego i oprogramowania, zwłaszcza w tak rozbudowanym pakiecie. Streszczając całą krytykę do minimum powiem, że odczuwam zbyt mało tak ducha amigowego, jak i magellanowego w całości i za dużo zapożyczeń z systemu Windows. Samoistnie nasuwa mi się skojarzenie, że autor mocno chciał odwzorować Windowsy w środowisku amigowym. Dostrzegam pewien brak konksekwencji, typu zbyt małe czcionki i ikonki w stosunku do dużych rozdzielczości pod emulatorem. Nie podoba mi się tez rezygnacja z funkcji wbudowanych w Magellana, jak np. zegarek czy Start menu, na rzecz zewnętrznych programów. Odnoszę wrażenie, że amigowy system zaproponowany przez autora powinien być raczej uruchamiany bez Magellana, raczej w trybie klasycznego Workbencha. Miałoby to większą rację bytu.



Wizualnie oceniam całość na piatkę. Na pewno AmiKit 7 pozytywnie zaskoczy kogoś niezaznajomionego wcześniej z Amigą wyposażoną w kartę graficzną. Za wykorzystanie "mocy" Magellana II i konfigurację amigowego desktopu z jego pomocą dam co najwyżej czwórkę minus. Chwała autorowi za wysiłek... I czekam na bardziej amigową w klimacie i bardziej funkcjonalną pod kątem wykorzystania Magellana, wersję 8.

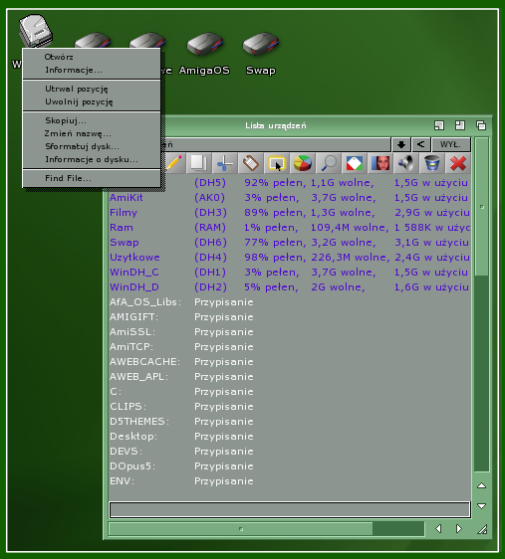
Dariusz Gac

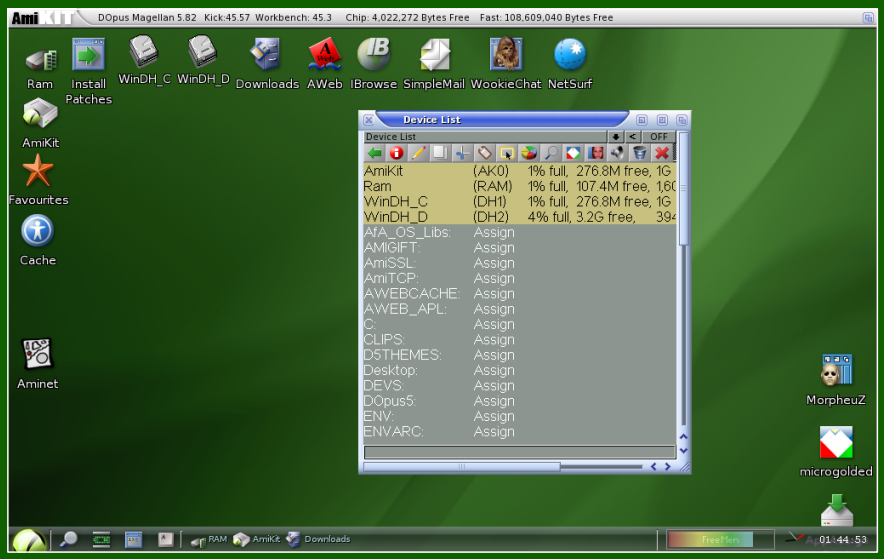
Strona projektu:

http://www.amikit.amiga.sk

- bogaty wybór i konfiguracja amigowego oprogramowania,
- dopracowane wizualnie ustawienia,
- prosta i intuicyjna instalacja.

- wymaga kickstartu 3.1 i systemu AmigaOS od 3.5 wzwyż,
- zbyt duże nawiązanie do systemu Windows,
- niepełne wykorzystanie możliwości Magellana,
- zdecydowany przerost formy.





Z lewej strony listery Magellana II w wersji oryginalnej, z prawej po drobnych poprawkach.

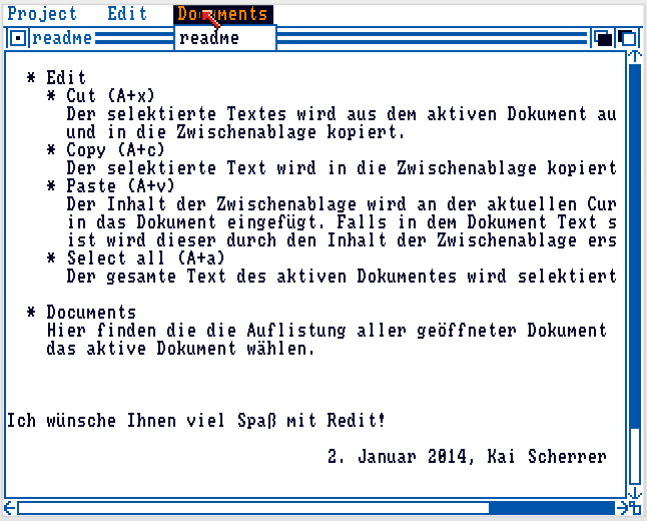
Redit

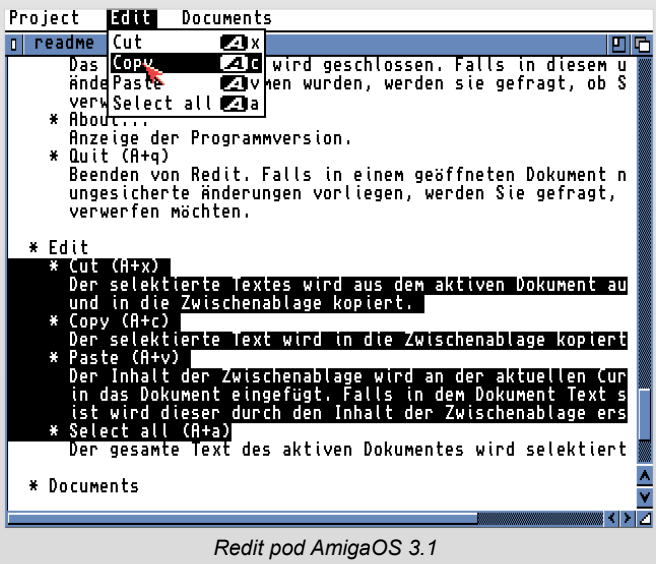
Edytorów tekstu Amiga klasyczna doczekała się wielu. Niektóre są lepsze, inne gorsze, ale wszystkie mają jedną podstawową zaletę, z którą zgodzą się chyba wszyscy – są lepsze niż systemowy *Ed*. Ale czy to zdanie jest na pewno prawdziwe w przypadku bardzo ortodoksyjnych użytkowników Amigi, którzy nie wyściubiają nosa spod Amig działających pod AmigaOS 1.2? Nie za bardzo, gdyż tacy użytkownicy raczej nie mają alternatywy. Znalezienie dobrego edytora tekstu, który byłby w stanie zaspokoić ich podstawowe potrzeby mogło być trudne, o ile nie całkowicie niemożliwe. Aż do teraz, gdyż Kai Scherrer napisał program o brzmiącej z niemiecka nazwie *Redit*.

Nasz bohater to prosty edytor plików tekstowych. Zajmuje niewiele miejsca (plik wykonywalny liczy sobie niecałe 60 kB) i ma niewyobrażalnie niskie wymagania – zadowoli się praktycznie każdą Amigą z systemem od AmigaOS 1.2 aż po AmigaOS 3.9, z 0,5 MB pamięci. Podobnie jak i wymagania, jego możliwości są również niewielkie, co jednak nie znaczy, że całkowicie eliminują program z centrum zainteresowania. Jeżeli zaczniemy go porównywać na przykład z *CED-em*, wówczas szybko dojdziemy do wniosku, że takie porównanie nie ma sensu (choćby na sam start – brak jakiejkolwiek konfiguracji), ale już myśląc w kategoriach zastąpienia nim *Ed-a*, ma to jak najbardziej szansę powodzenia. Powodów jest kilka, z czego najważniejsze to intuicyjna, w pełni amigowa obsługa (mówcie sobie co chcecie, ale moim zdaniem *Ed* jej nie ma), podstawowe funkcje kopiowania, wycinania i wklejania tekstu czy praca na wielu dokumentach otwartych jednocześnie w różnych oknach. O takich elementach, jak odczyt i zapis pliku nie ma co wspominać, gdyż są one absolutnym minimum, jakie każdy program powinien posiadać. Dodatkowo, wszystkie funkcje programu można wywołać korzystając ze skrótów klawiaturowych, których obłożenie jest przyjęte jako standard (pod kombinacją klawiszy Amiga+C mamy kopiowanie, a nie na przykład wycinanie).

Redit pod AmigaOS 1.3

Instalacja programu, podobnie jak i jego wykorzystanie są banalnie proste. Jeden plik, który należy skopiować w wybrane miejsce na dysku i uruchamiać w przypadku zaistnienia takiej potrzeby. Nie ma konieczności dogrywania żadnych bibliotek ani dodatkowych komend, chociaż w przypadku ich dostępności lub lepszej konfiguracji systemu, program to zauważa i robi z tego użytek (np. tak działa procedura schowka, która po wykryciu *clipboard.device* robi z niego użytek zamiast wykorzystywać wbudowane mechanizmy programu). Program startuje zarówno z poziomu CLI, jak i Workbencha zapewniając użytkownikowi swobodę w zakresie jego wywołania.





Redit pod AmigaOS 3.1

Zapewne rodzi się pytanie, dla kogo jest taki, dość ubogi, ale jednocześnie bardzo przyjazny systemowi, o bardzo niskich wymaganiach program? Osobiście zastosowania znalazłem dwa. Pierwsze z nich to, już wspomniane wcześniej, zastąpienie „koślawego", nieintuicyjnego i kiepskiego *Ed-a*, co na pewno pozytywnie odbije się na zdrowiu psychicznym wielu z Was, którzy są czasami zmuszeni, aby z niego korzystać. Drugie, jakie wpadło mi do głowy, to wykorzystanie programu przy wszelkiego rodzaju pakietach startowych dla „początkujących". Takie zastosowanie program już zresztą znalazł, gdyż znajduje się w opisywanym choćby w tym numerze *BetterWB*. Szybko zbierając pomysły, wydaje mi się, że można go również wykorzystać przy dyskietkach ratunkowych lub dyskietkach startowych, gdzie niezbędne jest użycie edytora tekstu, a z różnych przyczyn (miejsce na dysku, wolna pamięć, ograniczenia systemowe) wybór mamy bardzo zawężony. Z całą pewnością amigowe wygi, o ile jeszcze w ogóle korzystają z edytorów tekstu, nie zainteresują się tym programem, gdyż nie przedstawia on dla nich żadnej wartości. Mała liczba funkcji i prostota mogą się okazać zbyt dużymi ograniczeniami, które zdeklasują w ich oczach ten program.

Sebastian Rosa

Strona projektu:

http://www.amikit.amiga.sk

- niewyobrażalnie niskie wymagania – program działa bez problemów nawet na AmigaOS 1.2,
- podstawowe funkcje edycji tekstu, takie jak kopiowanie, wklejanie, wycinanie,
- możliwość pracy na kilku dokumentach jednocześnie, otwartych w osobnych oknach.

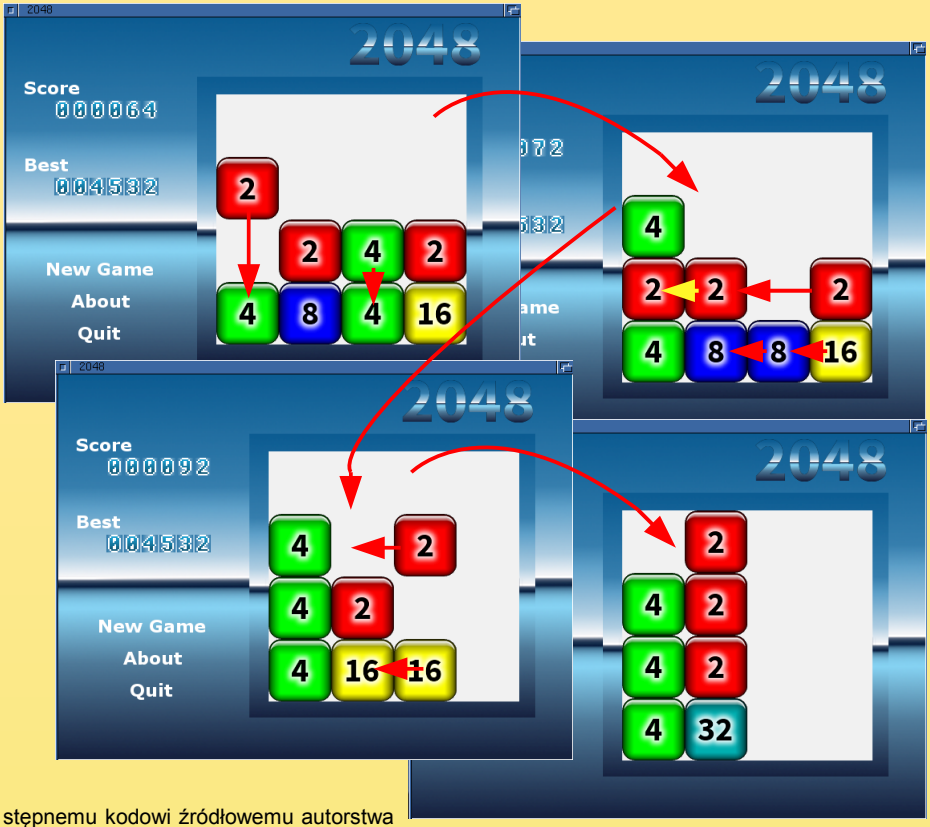
- brak jakiejkolwiek, choćby podstawowej możliwości konfiguracji.

2048

Jeszcze kilka lat temu cały świat zagrywał się w *„Sudoku”* – czy to w postaci papierowej, czy elektronicznej, choć było to nieistotne, gdyż każda z dostępnych odmian pozwalała skutecznie zabić czas w drodze do pracy/szkoły czy też podczas śniadania, jednocześnie cały czas ćwicząc nasze szare komórki. W dobie coraz bardziej zaawansowanych telefonów komórkowych i tabletów gry logiczne powoli odchodzą w zapomnienie i wypierane są przez gry bardziej zręcznościowe: *„Angry Birds”*, *„Flappy Bird”* – te tytuły nie schodziły z wokandy pulpitów „tableciarzy” na całym świecie w okresie ostatnich kilku latach. Rok 2014 to chyba jednak renesans gier logicznych, gdyż furorę robi między innymi tytuł *„2048”*. Z jednej strony gra niezwykle prosta, zarówno w obsłudze, jak i w zasadach w niej obowiązujących, a z drugiej strony dosyć wymagająca i przy tym jednocześnie wciągająca.

Gracz ma przed sobą kwadratową planszę składającą się z 16 pól oraz dwa znajdujące się na niej w losowych miejscach klocki z cyfrą 2 i/lub 4. Zadaniem gracza jest przesuwanie klocków po planszy przy wykorzystaniu klawiszy sterujących kursora. Należy jednak mieć na uwadze, że nie przesuwamy każdego klocka z osobna, lecz ich całe wiersze lub kolumny aż do momentu, gdy napotkają przeszkodę, którą jest kolejny klocek lub granica planszy. Każde przesunięcie skutkuje dwoma zdarzeniami. Pierwsze z nich występuje zawsze i jest to pojawienie się na dowolnym, pustym polu planszy nowego klocka z cyfrą 2 lub 4. Drugie jest warunkowe i zależy od tego czy dwa klocki z tą samą cyfrą lub liczbą zderzą się. Jeżeli miała miejsce taka sytuacja dwa klocki z tą samą cyfrą lub liczbą łączą się w jeden o wartości będącej ich sumą. W ten sposób następuje systematyczne zwiększanie wartości klocków, aż do pomyślnego uzyskania tytułowej liczby 2048. W trakcie rozgrywki wszystkie wartości połączonych klocków zwiększają również ogólny wynik punktowy gracza.

Popularność tej dosyć prostej, lecz niezwykle wciągającej gry sprawiła, że dzięki ogólnodostępnemu kodowi źródłowemu autorstwa Gabriele Cirulli, niczym grzyby po deszczu, zaczęły pojawiać się jej różne inkarnacje na różne systemy operacyjne. Robert Krajcarz wraz z Aleksandrem Piotrem Chylińskim stworzyli także jej amigową odmianę. Jak na takie maleństwo, gra ma dosyć spore wymagania – konieczny jest system AmigaOS 3.x lub wyższy (także system AmigaOS 4.x, jak i MorphOS) oraz datatyp PNG. Jak to się już przyjęło w przypadku dokonań Roberta i Aleksandra, gra działa komfortowo na kartach graficznych, jak i na chipsecie AGA (nie było testowane działanie na chipsecie ECS), zarówno w trybie pełnoekranowym, jak i w okienku oraz posiada tzw. oskórowanie, a więc kilka różnego rodzaju zestawów graficznych, w zależności od upodobań gracza lub konfiguracji posiadanego sprzętu (wybór skórek odbywa się przez parametr/tooltype *SKIN*). Dodatkowo, gra zapisuje jeden, najlepszy uzyskany wynik punktowy.



Wizualna prezentacja zasad gry – linie proste wskazują kierunek ruchu gracza i efekt, jaki to za sobą niesie. Linie krzywe, to kolejność wykonywanych kroków.

http://blabla.ppa.pl/download/2048.lha

Gra jest prosta, napisana przyjaźnie dla systemu i w zasadzie nie ma sensu się nad nią zbyt rozpisywać. Cieszyć powinien fakt, że to, w co zagrywa się cały świat, możemy mieć również i my na naszych poczciwych, starych (a także i nowych) Amigach lub ich wcieleniach. Nie powinniśmy również mieć żadnych pretensji do jakości wykonania tej amigowej odmiany gry, gdyż Robert Krajcarz i APC nie raz już udowodnili, że stanowią pewnego rodzaju gwarancję tego, iż ich produkt zawsze posiada wystrój adekwatny do swojego charakteru (choć niekoniecznie emanuje wodotryskami graficznymi) oraz, co chyba najważniejsze, jest zawsze ukończony lub ukończony na tyle, aby kolejne wersje były tylko usprawnieniami poprzedniej. Panowie, tak trzymać i czekamy na Waszą kolejną „małą” grę.

Sebastian Rosa

4096

„4096” to wariacja oryginału napisana przez Roba Hewitta z grupy Coagulus. Zasady są dokładnie takie same, lecz autor zwiększył poziom trudności podnosząc dwójkę do kolejnej potęgi, dopiero wtedy umożliwiając graczowi zakończenie zabawy. Graficznie gra nie wygląda może rewelacyjnie (paleta kolorów jest trochę matowa), lecz dzięki temu tytuł nie posiada zbyt dużych wymagań i zadziała na Amidze z 1 MB pamięci. Rozgrywka jest dokładnie taka sama.

http://coagulus.newport.net/cs/bin/4096.zip

- atrakcyjny wygląd,
- poprawność wykonania technicznego,
- nowość na świecie, nowością na Amidze.

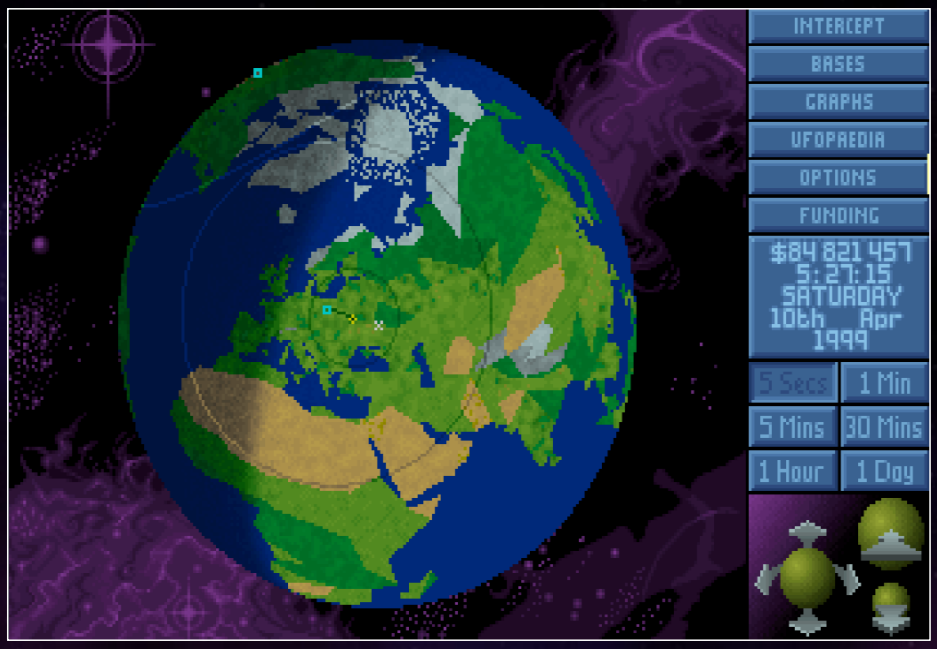
- drobne przekłamanie graficzne wersji AGA (zalecane jest granie na własnym ekranie, a nie w okienku).

OpenXcom

„Ufo: Enemy Unknown” to gra, która została wydana w 1994 roku. Niestety, tytuł, który bez problemów działał nawet na poczciwej Amidze 500, sprawia problemy na nowych komputerach. Dlatego też grupa fanów tej gry postanowiła napisać wersję, która zadziała bez problemów na nowych komputerach. W ten sposób narodził się w 2009 roku projekt *OpenXcom*. Pierwsza wersja gry ukazała się w październiku 2010 roku, a dzięki dużym wysiłkom Pawła „Stefkosa” Stefańskiego od niedawna możemy cieszyć się portem tej gry także dla systemu MorphOS.

Historia gry jest prosta – Obcy z innych systemów gwiezdnych pojawiają się na naszej planecie i zaczynają terroryzować ludzkość. Powołana zostaje tajna organizacja X-Com, która postawiła sobie jeden cel – zniszczyć najeźdźców. Podstawowa rozgrywka nowej odsłony klasyka jest dokładnie taka sama, jak pierwowzoru i w tym artykule nie będziemy się na tym skupiać. Wiele już o *„UFO: Enemy Unknown”* napisano i zainteresowani na pewno znajdą wiele informacji w szeregu popełnionych recenzji, znajdujących się także na naszym portalu. W tym tekście postaram się przyjrzeć bliżej temu, jakie zmiany wnosi nowa inkarnacja gry, bo jak to mówią – diabeł tkwi w szczegółach.

Najważniejsza z nich jest taka, że *OpenXcom*, dzięki wieloplatformowej bibliotece SDL, przy pomocy której została napisana, bez najmniejszego problemu można uruchomić na systemie MorphOS lub Windows. Wystarczy pobrać odpowiednie archiwum, dograć pliki z oryginalnej wersji gry i już można zaczynać zabawę. Jednak nie to jest największą zaletą *OpenXcom* – siła gry tkwi w możliwości zastosowania modyfikacji, które mogą całkowicie zmienić rozgrywkę. *OpenXcom* udostępnia nam cały szereg zmian, które uprzyjemniają rozgrywkę. I tak w wersji gry dla systemu

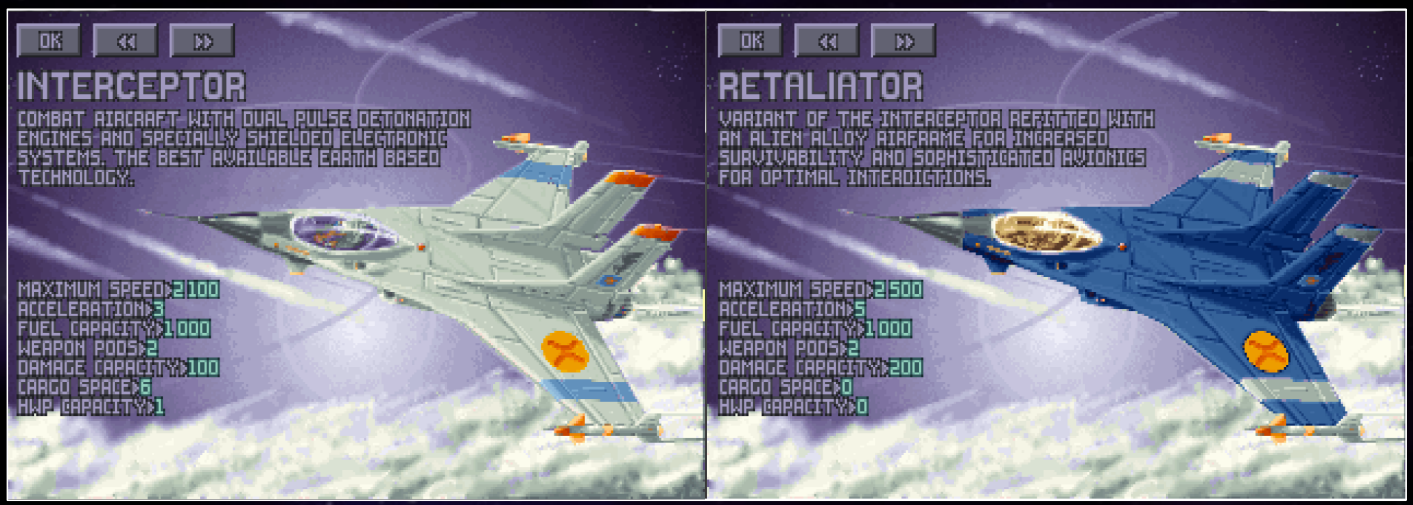


MorphOS możemy:

- zmienić rozdzielczość – u mnie wybór jest od 640x480 aż do 1600x900,
- wybrać tryb wyświetlania – okienko lub pełny ekran,
- nałożyć filtry na obraz, które mają poprawić jakość obrazu (choć nie zawsze tak jest),
- wybrać wersję językową – jest też dostępny język polski,
- zmienić zasady gry w opcjach zaawansowanych.

Tych ostatnich, które w przypadku *OpenXcom* odgrywają największą rolę, jest bardzo dużo i pozwalają one między innymi na następujące atrakcje:

- agresywne akcje odwetowe, czyli UFO zawsze szuka naszych baz,
- natychmiastowe granaty, czyli granat eksploduje zaraz po rzucie,
- podstępna AI – UFOki trochę inaczej się zachowują, preferują ostrzał z odległości, przez co częściej używają np. Blaster Launchera,
- ustawienie rozmiaru eksplozji, czyli jak wysoko w górę sięga detonacja (ja mam ustawione na 3 pola, przez co zabawa w mieście nabiera dodatkowych rumieńców, gdyż wystarczy, że obcy chowa się na piętrze, a ja podkładam C4 na parterze i bum),
- sprzedaż żywych obcych – żywy jest wart więcej niż martwy, więc „nadwyżkę” inwentarza można sprzedać,



Z lewej strony klasyczny Interceptor, a z prawej jego delikatnie zmodyfikowana wersja, która otrzymała nazwę Retaliator.

- przymusowy start – można zmusić do startu nasze myśliwce, nawet jeżeli się nie przezbroiły/uzupełniły paliwa/są uszkodzone.

Ustawień jest znacznie więcej – wymieniłem jedynie te, które wydają mi się na pierwszy rzut oka najciekawsze. Dodam, że ja przez prawie tydzień dobierałem różne konfiguracje, aż znalazłem „tę idealną". Jednak to i tak nie jest wszystko, co *OpenXcom* ma do zaoferowania. Najważniejsze w całej zabawie są mody, które dodają np. nowe uzbrojenie, nowe myśliwce.

Mody można pobrać ze strony autorów portu. Archiwum rozpakowujemy i pliki wgrywamy do katalogu Resources, który znajduje się w katalogu z główną grą. Ja osobiście polecam zainteresować się:

- Combat Armour – pancerz, lepsze to niż mundurek, ale słabsze od Personal Armour,
- Combat Knife – nóż, którym zabijesz nie jednego Sectoida,
- Firestorm Restyle – inna skórka myśliwca Firestorm,
- Grenade Launcher – granatnik, lżejszy od bazooki a równie skuteczny,
- Light Machinegun – karabinek, zaskakująco skuteczny na bliskie odległości,
- Retaliator – ulepszony Interceptor,
- Sniper Rifle – snajperka, co tu dużo pisać, jeden strzał i Sectoid trupem leży.

Po tych kilku modach, *OpenXcom* zmienił pierwowzór w całkowicie nową grę – niby jest to stare *„UFO: Enemy Unknown"*, a jednak gra się zupełnie inaczej. Obcy są sprytniejsi, potrafią zasadzić się z Blasterami i wytłuc całą ekipę ziemskiej federacji. *„UFO: Enemy Unknown"* to tytuł, który pomimo tego, że koncepcyjnie ma 20 lat, wcale się nie starzeje, a dzięki tej rewitalizacji, gra odmłodniała. Rozgrywka jest równie przyjemna, jak dwie dekady temu, o ile nawet nie lepsza.

Nie ma jednak róży bez kolców – nawet taki ideał jak *OpenXcom* ma swoje wady. Największa z nich to prędkość działania wersji dla systemu MorphOS. Jest niestety WOLNO. Na moim G4 1.5 GHz i karcie graficznej Radeon 9600PRO, zajętość procesora przez cały czas utrzymuje się na poziomie 100%, a prędkość pozostawia wiele do życzenia. Nie jest to wina wolnego procesora, a niedbale napisanego kodu i efekt wykorzystania SDL. To wszystko sprawia, że wersja dla systemu MorphOS działa wolniej niż powinna i dużo wolniej niż analogiczna wersja przeznaczona dla systemu Windows, która po prostu śmiga, choć często zdarza się jej „wykrzaczyć". Być może jest to spowodowane wykorzystaniem któregoś z modów, a być może i samej. Oprócz tej małej niedogodności, gra nie posiada praktycznie wad. Oprócz może jeszcze jednej – wciąga. Podczas pisania tego artykułu uruchomiłem „na chwilę" UFO i okazało się, że ta chwila trwała prawie dwie godziny. Wiadomo, planeta sama się nie uratuje.

Aleksander „trOLLO" Giedyk

Oficjalna strona projektu:
http://openxcom.org

Wersja dla MorphOS-a:
http://stefkos.amigazeux.net/openxcom.lha

Zasadniczo to stare, dobre „UFO", ale mody pozwalają wejść w posiadanie nowego wyposażenia naszych baz oraz żołnierzy.

Nasza pierwsza gra

Kurs programowania z wykorzystaniem Visual Studio 2005 Express Edition i VBCC – część 8

W ósmej odsłonie tworzenia gry skupimy się na następujących rzeczach: ulepszymy menu w obrazku tytułowym, które zaczęliśmy w poprzednim odcinku, dodamy możliwość konfigurowania klawiszy poprzez menu Options oraz uatrakcyjnimy rozgrywkę umieszczając licznik punktów w dolnym panelu.

Aby zająć się porządnie konfiguracją klawiszy, musimy zrobić odpowiednie menu z możliwością edycji pól tak, aby były widoczne zmiany dokonane przez gracza. W siódmej części w **title.c** w bardzo prosty sposób napisaliśmy takie przykładowe menu. Niestety nie jest ono wystarczająco elastyczne dla naszych nowych potrzeb, więc niewiele myśląc zmienimy je. Początkującym twórcom takie zmiany mogą się kojarzyć z niewłaściwą drogą, bo przecież wystarczy tylko nieznacznie rozbudować to, co zaczęliśmy robić, dodając brakujące elementy, a to, co już jest zostawić, skoro zostało to już przetestowane. Ja uważam inaczej z dwóch powodów. Po pierwsze, ulepszanie przez rozbudowanie przy złym podejściu może skutkować zawiłym kodem, a po drugie warto programować na różne sposoby (a nuż wymyśli się coś zupełnie nowego). W każdym razie tworzymy nowe menu. Standardowo dodajemy nowy moduł menu do naszego projektu. Myślę, że nie sprawi to problemu czytelnikowi, bo temat ten przewijał się w paru odcinkach. Najważniejszą rzeczą w menu będzie odpowiednia struktura, która ułatwi nam życie zarówno przy wyświetlaniu, jak i w poruszaniu się po menu. Będzie to struktura odpowiedzialna za jedną linię menu.

```
typedef struct
{
        int posX;
        int posY;
        char* name;
        int color;
} MenuEntry;
```

Widzimy, że jest to bardzo intuicyjne podejście. Mamy tu pozycję poziomą, pionową, kolor i ciąg znaków do wyświetlenia. Umieszczenie koloru na końcu struktury ma znaczenie praktyczne – służy zmniejszeniu ilości błędów podczas programowania oraz nie musimy się zastanawiać czy najpierw jest kolor a potem pozycje x i y, czy też najpierw są pozycje a później kolor. Mając takie *MenuEntry* możemy teraz sobie stworzyć przykładowe menu, robiąc odpowiednią tablicę, którą umieściłem w **title.c**.

```
static MenuEntry m_menuTitle[] =
{
    {MENU_POS_X, MENU_START_POS_Y, "start", COLORTXT_LIGHT},
    {MENU_POS_X, MENU_OPTIONS_POS_Y, "options", COLORTXT_NORMAL},
    {MENU_POS_X, MENU_EDITOR_POS_Y, "editor", COLORTXT_NORMAL},
    {MENU_POS_X, MENU_EXIT_POS_Y, "exit", COLORTXT_NORMAL},
};
```

Nietrudno się domyślić – *MENU_POS_X* i *MENU_START_POS_Y* to przykładowe pozycje zdefiniowane za pomocą dyrektywy *#define*. Słowo komentarza należy się kolorom. Aby menu na starcie wyglądało jak należy, pierwsza jego linia powinna mieć inny kolor niż pozostałe, aby było wiadomo, że zaczynamy od tego miejsca. Zastanówmy się teraz, w jaki sposób będziemy realizować przemieszczanie się po menu. Oczywiście będziemy się poruszać za pomocą joysticka lub klawiatury. Oprócz tego nasza gra nabrałaby kolejnych rumieńców, gdyby można było użyć do tego celu również myszki. Tym zajmiemy się chwilę później. Jak wiemy, jeden kolor mamy inny w tablicy *m_menuTitle* i jeśli będziemy odpowiednio nim manipulować, umieszczając kolor *COLORTXT_LIGHT* w różnych wierszach, to rozwiążemy problem poruszania się po menu. Co więcej – nasze menu zawiera też informacje o ograniczeniach, czyli kiedy gracz już nie może iść do góry czy też do dołu (mówią o tym pozycje pionowe w pierwszej i ostatniej linii w tablicy). Załóżmy, że wystąpił ruch w naszym menu – wtedy musimy na samym początku zamienić kolor ówczesnego podświetlenia na normalny kolor, czyli innymi słowy musimy zgasić podświetloną wcześniej pozycję w menu. Potem trzeba tylko podświetlić napis w nowej pozycji i gotowe. Oto funkcja realizująca to zadanie.

```
void MenuUpdate(MenuEntry* menuTable, int amount, int nPosY)
{
        int i = 0;
        while (i < amount)
        {
                if (COLORTXT_LIGHT == menuTable[i].color)
                {
                        break;
                }
                i++;
        }
        menuTable[i].color = COLORTXT_NORMAL;
        const MenuEntry entry = menuTable[i];
        PrintTxt(entry.name, entry.color, entry.posX, entry.posY);
        for (i = 0; i < amount; ++i)
        {
                if (nPosY == menuTable[i].posY)
                {
                        break;
                }
        }
        menuTable[i].color = COLORTXT_LIGHT;
        const MenuEntry e = menuTable[i];
        PrintTxt(e.name, e.color, e.posX, e.posY);
}
```

Samo wyświetlenie całego menu jest banalne i nie będę go przedstawiał, gdyż sprowadza się to do jednej pętli, w której wyświetlamy napis o znanej pozycji i kolorze. Ponieważ wcześniejsza wersja funkcji odpowiedzialnej za pokazanie tekstu na ekranie, znajdująca się w module *tile*, nie miała możliwości zmiany koloru, to ona także została napisana od nowa. Aby ułatwić sobie pracę z tekstem, dodałem też dwie dodatkowe wersje funkcji pokazujące tekst w normalnym i podświetlonym kolorze. Pojawiła się też nowa funkcja – *BackgroundToBlack*, której zadaniem jest zrobienie czarnego tła w całym oknie. Oprócz wyświetlenia menu będziemy potrzebować też funkcję inicjującą menu, która przechodząc w pętli wszystkie pozycje menu ustawi podświetlenie na pierwszej.

Jak zaznaczyłem już wcześniej, dodamy obsługę myszki w naszym menu. Zaczniemy najpierw od umieszczenia niezbędnych rzeczy w module *window*. Tam bowiem będą zbierane eventy o przyciskach urządzenia i tam też dołączymy funkcje zwracające pozycje poziomą i pionową myszki. W każdym razie powiększymy ilość tagów okna, dorzucając *WA_RMBTrap*, który blokuje dostęp do menu (mam tu na myśli główny pasek na samej górze w Workbenchu) dla prawego przycisku naszego elektronicznego stworzenia, a także dopisujemy *IDCMP_MOUSEBUTTONS* w tagu *WA_IDCMP*, umożliwiając systemowi generowanie zdarzeń o przyciskach. Oczywiście musimy umieścić odpowiedni blok kodu, który przeanalizuje zdarzenie *IDCMP_MOUSEBUTTONS*, a sam kod polega na sprawdzeniu odpowiednich stałych w *msg_code*, takich jak *SELECTUP* – dla wciśniętego lewego przycisku myszki, *SELECTDOWN* – dla puszczonego. Podobnie będzie dla prawego przycisku – tu mamy *MENUUP* i *MENUDOWN*. W **window.c** dokładamy funkcje *GetMouseY* i *GetMouseX* zwracające pozycję poziomą i pionową myszki, które pobieramy odpowiednio z pól *GZZMouseX* i *GZZMouseY* ze struktury *Window*. Tak uzbrojeni możemy przystąpić do właściwej procedury zawierającej obsługę menu wraz z pozycją y myszy. A zatem na samym początku pobieramy wspomnianą pozycję pionową. Aby dobrać się do linii w menu, musimy, tak na chłopski rozum, pozycję podzielić przez wysokość fonta, wziąć część całkowitą i pomnożyć przez wysokość fonta. Dzięki temu otrzymamy jedną pionową pozycję w obrębie danej linii menu. Ja zastosowałem tutaj sztuczkę, aby uniknąć dzieleń i mnożeń. Bo jak pamiętamy – w naszej grze wysokość fontów wynosi 8 pikseli. Te 8 pikseli w zmiennej *posY* zajmuje 3 pierwsze bity (licząc oczywiście od lewej). A więc, aby osiągnąć jedną pozycję w linii, wystarczy te 3 bity skasować w zmiennej. I dokładnie to robimy wykorzystując do tego celu negację i koniunkcję (and). Wprowadzamy też zmienną przetrzymującą starą wartość pozycji *m_nOldPosY*, aby było możliwe usunięcie starego podświetlenia menu oraz ze względów wydajnościowych (przy małych ruchach myszki nie ma potrzeby aktualizacji podświetlania). Jeśli faktycznie użytkownik wykonał ruch w kierunku zmiany linii w menu, to wówczas pozostaje nam sprawdzić nasze menu i pozycję pionową, aby wiedzieć którą linię musimy zaktualizować.

```
static WORD m_nOldPosY = 0;

void MenuMousePosY(MenuEntry* menuTable, int amount)
{
   const WORD posY = GetMouseY() & ~0x07;
   if (m_nOldPosY != posY)
   {
     int i;
     m_nOldPosY = posY;
     for (i = 0; i < amount; ++i)
     {
       const MenuEntry entry = menuTable[i];
       if (posY == entry.posY && COLORTXT_NORMAL == entry.color)
       {
         MenuUpdate(menuTable, amount, posY);
         ArrowUpdateY(posY);
         break;
       }
     }
   }
}
```

W powyższym bloku kodu pojawiła się tajemnicza funkcja *ArrowUpdate*, która zajmuje się aktualizacją strzałki widocznej z lewej strony. Cały wcześniejszy kod dotyczący obsługi strzałki został przeniesiony z modułu *title* do *arrow* i właśnie nim zajmiemy się teraz. Najważniejszą funkcją jest *ArrowInit*, której zadaniem jest inicjalizacja niezbędnych rzeczy do poprawnego funkcjonowania strzałki. Są to po-

zycja pozioma i pionowa a także granice – dolna i górna. Oprócz tego zerujemy licznik opóźniający animację i ustawiamy początkowy znak z fontów, który przedstawia graficzny symbol strzałki. Pozostałe funkcje to: *ArrowAnimate*, czyli serce naszego modułu, służące do animacji strzałki, *ArrowDown* i *ArrowUp* przemieszczają znak do dołu bądź do góry o jedną linię, która ma stałą wielkość 8 pikseli. Za pomocą funkcji *ArrowGetPosY* możemy z innego modułu dowiedzieć się o aktualnej pozycji pionowej, a *ArrowUpdateY* aktualizuję pozycję y.

W module *options* znajdziemy menu odpowiedzialne za opcje w naszej grze. W poprzednim odcinku dodaliśmy konfigurację dla klawiszy i kolejnym bardzo naturalnym krokiem jest właśnie możliwość ich zmiany. Modyfikacja klawiszy w dowolny sposób musi pociągać za sobą odzwierciedlenie w postaci napisu, jaki to klawisz użytkownik wybrał, a zatem potrzebujemy tablicy zawierającej wszystkie możliwe klawisze. Przedstawię kawałek takiej tablicy, bo zwyczajnie szkoda miejsca na takie tabelki.

Na marginesie warto wiedzieć, jaki kod klawisza przychodzi w zdarzeniu *IDCMP_RAWKEY*. Jest on w *msg_code*. Siedem młodszych bitów mówi o klawiszu, dla przykładu ESC ma kod 0x45, a ostatni 7 bit (licząc od zera) mówi o tym czy klawisz został naciśnięty. Jeśli gracz wciśnie i puści ESC, to zostaną wygenerowane dwa zdarzenia *IDCMP_RAWKEY* i *msg_code* będzie zawierał 0xC5 (0x45+0x80), a później będzie to 0x45.

```
const char *lv_all_keys_tab[] = {
  "`",          /*;00 */
  "1",          /*;01 */
  "2",          /*;02 */
  "3",          /*;03 */
  "4",          /*;04 */
```

Mamy tu tablicę zawierającą wskaźniki do napisów i poruszając się po indeksach tejże tablicy mamy napis odpowiadający klawiszowi. Jeśli gracz wciśnie na klawiaturze jedynkę, to odpowiada to pierwszemu (liczymy od zera w tym przypadku) elementowi naszej tablicy. Ułożenie napisów w tej tablicy nie jest przypadkowe – jest ono zgodne z kodami otrzymywanymi za pomocą zdarzenia *IDCMP_RAWKEY*. Swoją drogą, zbieraliśmy tylko niektóre kody klawiszy w **window.c** – pora to zmienić. Najpierw definiujemy w module *input* tablicę przechowującą wszystkie klawisze – będzie ich aż 128. Oczywiście klawiszy na klawiaturze jest mniej, ale na 7 bitach tyle można pomieścić wartości.

```
UBYTE nKeyValue = 1;
if (msg_code & 0x80)
{
    nKeyValue = 0;
}
g_bKeys[msg_code & 0x7f] = nKeyValue;
```

Powyższy kod wstawia do tablicy jedynkę o indeksie równym kodowi klawisza – o ile gracz nacisnął klawisz. Jeśli ten sam klawisz został puszczony, to wówczas w to miejsce powędruje zero. W **input.c** doszły dwie nowe funkcje wspierające klawiaturę: *ClearKeysTable* czyści tablicę przechowującą kody klawiszy a *GetFirstPressedKey* zwraca pierwszy klawisz, jaki został wciśnięty, a jeśli zostało wciśniętych kilka klawiszy, to zostanie zwrócona wartość odpowiadająca najniższemu kodowi klawisza. Czyli jeśli gracz wcisnął jednocześnie klawisze 3 i ESC, to po pierwsze przyjdą dwa zdarzenia *IDCMP_RAWKEY* i dla każdego inny indeks tablicy klawiszy zostanie ustawiony na jeden, a *GetFirstPressedKey* zwróci kod odpowiadający klawiszowi 3, czyli w tym przypadku właśnie 3. A co jeśli żaden klawisz nie został wciśnięty? Co w takim przypadku funkcja zwróci? Jasne jest, że nie może zwrócić poprawnej wartości z zakresu 0-127. Oznaczałoby to, że jednak klawisz został wciśnięty – przyjąłem, że będzie to 255. Mając odpowiednie funkcje w module *input*, możemy zabrać się za napisanie pętli głównej w menu *options*. Najważniejsze w tej pętli to oczekiwanie na przycisk czy to joysticka, czy to myszki oraz czekanie na ruch dół/góra. Jedyne co jest zawsze robione w ciele funkcji, to animacja strzałki. Szybko omówmy przypadki łatwiejsze, które to w zasadzie aktualizują menu zgodnie z tym, co gracz zrobił. Jeśli skierował joystick do góry, to strzałka wędruje do góry i aktualizujemy podświetlanie menu. Bardzo podobnie to wygląda dla wychylenia w dół kontrolera. Słowo wyjaśnienia dlaczego korzystamy z konstrukcji *if else* zamiast używać tylko *if*. Naiwnie rozumując, ruch myszki i ruch joystickiem są niezależne od siebie i nie powinny generować problemu. Z początku też tak myślałem, ale szybko się okazało, że strzałka może być w innym miejscu niż podświetlenie menu i wygląda to co najmniej nieciekawie. Przejdźmy do przypadku, gdy użytkownik wcisnął przycisk klawiatury. Na początku odtwarzany jest dźwięk. Następnie pobieramy pozycję pionową strzałki, sprawdzamy której linii menu to dotyczy i jeśli udało się ją odnaleźć, to wyświetlamy komunikat proszący o wciśnięcie klawisza. Potem następuje przejście do procedury czekającej na puszczenie klawisza. Odbywa się to w typowy dla nas sposób – poprzez ustawienie wskaźnika na funkcję. Sama procedura czekająca, wykorzystuje do tego celu niepoprawną wartość 255, wykonując się w kółko aż żaden klawisz nie będzie wciśnięty. Po spełnieniu tego warunku czeka na wciśnięcie klawisza, pokazuje na ekranie ten klawisz w postaci napisu i powraca do pętli głównej w menu *options*.

```
static void optionsLoop(void)
{
            if (g_bFire || g_bMouseLeft)
            {
                        PlaySfx(SND_CLUNK);
                        g_bFire = FALSE;
                        g_bMouseLeft = FALSE;
                        int nPosY = ArrowGetPosY();
                        int i;

                        for (i = 0; i < ARRAY_SIZE(m_menuOpts) - 1; ++i)
                        {
                                    const MenuEntry key = m_menuOpts[i];
                                    if (key.posY == nPosY)
                                    {
                                                PrintLightTxt("press a key", KEY_POS_X, key.posY);
                                                m_nWaitPosY = key.posY;
                                                pNextFnc = &waitForKey;
                                                pCfgValue = (g_pCfg + i);
                                                g_pFnc = &wiatForReleasedKey;
                                                break;
                                    }
                        }
                        if (EXIT_POS_Y == nPosY)
                        {
                                    CopyCfgKeys(g_pCfg);
                                    g_bCfgChanged = TRUE;
                                    g_pFnc = &Title;
                        }
            }
            else if (g_bDown)
            {
                        g_bDown = FALSE;
                        ArrowDown();
                        MenuUpdate(m_menuOpts, ARRAY_SIZE(m_menuOpts), ArrowGetPosY());
            }
            else if (g_bUp)
            {
                        g_bUp = FALSE;
                        ArrowUp();
                        MenuUpdate(m_menuOpts, ARRAY_SIZE(m_menuOpts), ArrowGetPosY());
            }
            else
            {
                        MenuMousePosY(m_menuOpts, ARRAY_SIZE(m_menuOpts));
            }

            ArrowAnimate();
}
```

A na deser punktacja. Umieścimy ją na dole, a więc tym samym musimy powiększyć trochę okno zmieniając jedną zmienną w **window.c**. Poprzednia wersja gry wykorzystywała całą przestrzeń okna i nie było miejsca na punkty. Oprócz punktów na dole umieścimy też ilość żyć. Przeważnie w grach startuje się z wynikiem 0, a grając zdobywa się punkty. W naszym przypadku to standardowe podejście się nie sprawdzi, bo nie do końca wiadomo za co przydzielać punkty. Przecież nie można dostać punktów za to, że statek uderzył w kamień. Dlatego w naszej grze zrobimy na odwrót – gracz dostanie na starcie każdej planszy pewną ilość punktów, a grając punkty będą ubywać w zależności od tego, w jaki obiekt nasz statek uderzy. Wtedy to lepsi gracze, którzy odnajdą krótszą drogę, będą mieli więcej punktów. Logika punktacji powinna się znaleźć w module *game* i tam też umieszczamy zmienną *g_nScore*, za to odpowiedzialną. Do zwiększania i zmniejszania punktów posłużą dwie funkcje: *decreaseScore* i *increaseScore*.

```
static void decreaseScore(int amount)
{
            g_nScore -= amount;
            if (g_nScore < 0)
            {
                        g_nScore = 0;
            }
}
static void increaseScore(int amount)
{
            g_nScore += amount;

            if (g_nScore > 9999)
            {
                        g_nScore = 9999;
            }
}
```

W obu przypadkach sprawdzamy czy licznik punktów nie osiągnął ograniczenia. Ma to znaczenie przy wyświetlaniu punktów – będziemy wiedzieli ile minimalnie i maksymalnie miejsca zajmie punktacja. W tym przypadku będzie to najmniej jeden znak a najwięcej cztery. Skoro mówimy o pokazywaniu punktów, to w module *bottomPanel* umieściłem funkcję *PrintBottomPanel*, która między innymi rysuje wynik uzyskany przez gracza.

W dzisiejszym odcinku to tyle. Zachęcam jak zwykle do eksperymentowania z kodem i do zadawania pytań na forum PPA.

Asman

Zabawa obrazem w Pythonie

Raz na jakiś czas słyszymy o pewnych epokowych odkryciach. Dawno, dawno temu Kolumb odkrył Amerykę, potem Fleming trochę przypadkowo natrafił na sposób uzyskiwania penicyliny. Czasy teraźniejsze to również miejsce wielu genialnych odkryć zmieniających bieg historii każdego z nas. Mam tu na myśli chociażby zaprojektowanie i wyprodukowanie pierwszej Amigi czy odkrycie sposobu na pobieranie za frajer plików MP3 z chomikuj.pl. Przykłady można by mnożyć. Moim osobistym mega epokowym odkryciem ostatnich miesięcy jest biblioteka PIL umożliwiająca łatwe przetwarzanie plików graficznych za pomocą Pythona. I zanim dostanę nagrodę Nobla za to wyjątkowe odkrycie, podzielę się z wami całym dobrodziejstwem, jakie niesie za sobą użytkowanie PIL-a.

PIL, a raczej ***Python Imaging Library*** to moduł Pythona dostarczający szeregu bardzo przydanych funkcji do przetwarzania obrazów. Za jego pomocą możemy w banalny sposób (praktycznie za pomocą jednej metody) zmienić rozdzielczość obrazka czy obrócić go o dowolny kąt. Całą brudną robotę wykona za nas PIL. Oczywiście na tym nie kończą się jego możliwości. Dla ciekawskich polecam stronę internetową projektu. Tam znajdziecie pełną dokumentację wraz z przykładami do wszystkich metod składających się na moduł PIL.

„Prześlij przepis”

No dobrze, ale co ma to wspólnego z nami, amigowcami. A no tyle, że dzięki Yomguiowi, niezłomnemu krzewicielowi Pythona na platformie MorphOS, możemy cieszyć się biblioteką PIL na naszym ulubionym systemie. Archiwum z biblioteką w wersji **1.1.7-10** dostępne jest do pobrania z MorphOS Files. Tu od razu drobna uwaga. Poprzednia wersja biblioteki (1.1.7-8) zawiera błędy i żadnego chleba z niej nie upieczemy. Dlatego pobieramy najnowszą wersję, rozpakowujemy i kopiujemy część plików do katalogu *C*, a część do *LIBS:python2.5/site-packages* (zgodnie ze strukturą katalogów w archiwum PIL-a). Zakładam, że samego Pythona mamy już zainstalowanego. Jeśli nie, to szczegółową instrukcję znaleźć można w piątym numerze papierowego PPA (artykuł: *„Python pod MorphOS-em, czyli MP3 Renamer”*). Mając już zainstalowanego PIL-a, możemy trochę poszaleć. Za chwilę przedstawię prosty przepis na zmianę rozdzielczości przykładowego pliku graficznego w formacie JPG. Zatem do dzieła – odpalamy Scribble'a!

Przystawka

Po pierwsze musimy zaimportować moduł PIL tak, aby interpreter Pythona dowiedział się o istnieniu metod służących do obróbki plików graficznych. Rozpoczynamy nasz skrypt taką oto linią:

```
from PIL import Image
```

Następnie wskazujemy ścieżkę dostępu do pliku graficznego, który chcemy skonwertować. Funkcja ***Image.open*** zwróci nam obiekt typu *Image*, który ląduje w zmiennej o nazwie *picture*.

```
picture = Image.open("RAM:fotka.jpg")
```

Obrazek jest już wczytany do pamięci, możemy więc zmienić jego rozmiar za pomocą jednej z „magicznych” funkcji wujka PIL-a:

```
resized_picture = picture.resize((100,70),Image.ANTIALIAS)
```

w ten oto sposób obrazek kryjący się pod obiektem o nazwie *picture* został przeskalowany i ma teraz szerokość 100 px i wysokość 70 px. Najbardziej enigmatycznie może wyglądać użycie parametru ***Image.ANTIALIAS***. Jest to typ filtra, jaki zostanie użyty w trakcie przetwarzania pliku graficznego. Do wyboru mamy *NEAREST*, *BILINEAR*, *BICUBIC* oraz *ANTIALIAS*. Przy tym ostatnim otrzymywałem najlepsze jakościowo „obróbki”, dlatego też nie zawahałem się go użyć tutaj. Przeskalowany obrazek wędruje znowu jako obiekt typu *Image*, tym razem do zmiennej o nazwie *resized_picture*. Wystarczy go tylko zapisać jako „fotka_przeskalowana.jpg” z jakimś ludzkim współczynnikiem kompresji (np. na poziomie 85%, czyli z niewielkim, bo jedynie 15% ubytkiem jakości).

```
resized_picture.save("RAM:fotka_przeskalowana.jpg",quality=85)
```

Kompletny skrypt o nazwie *resize.py* wygląda następująco:

```
#!/usr/python
import sys
import os
from PIL import Image

picture = Image.open("RAM:fotka.jpg")
resized_picture = picture.resize((100,70),Image.ANTIALIAS)
resized_picture.save("RAM:fotka_przeskalowana.jpg",quality=85)
```

Zapisujemy skrypt do RAM, kopiujemy tam też plik *fotka.jpg* i odpalamy nasz garażowy konwerter plików JPEG z poziomu konsoli CLI za pomocą następującej komendy. Jeśli wszystko poszło dobrze (zgadzają się ścieżki do plików itd.), to po wykonaniu skryptu, RAM powinien wzbogacić się o nowy plik o nazwie *„fotka_przeskalowana.jpg”* o rozdzielczości 100x70 px.

```
python RAM:resize.py
```

Danie główne

Czas wykorzystać możliwości modułu PIL do jakiegoś zbożnego celu. Poniżej znajdziecie przykład bardzo przyjemnego skryptu pozwalającego na konwersję serii obrazów do dowolnej rozdzielczości. Do takich celów wręcz stworzona jest funkcja ***thumbnail***, która działa w bardzo podobny sposób do przedstawionej wcześniej metody ***resize***, z tą różnicą, że ***thumbnail*** czuwa nad zachowaniem odpowiednich proporcji przetwarzanego obrazu. Zatem możemy być pewni, że obrazki po przeskalowaniu nie zostaną nienaturalnie poszerzone albo zwężone. Na końcu podepniemy kompletny skrypt pod menu kontekstowe MorphOS-a tak, aby możliwe było hurtowe przeskalowanie obrazków (np. wskazanych w listerze Ambienta). Ale po kolei, najpierw zacznijmy od stworzenia nowej funkcji, którą nazwiemy *resize_picture*. Rzucimy jej na pożarcie trzy parametry:

- ***max_width*** – maksymalna szerokość obrazka,
- ***max_height*** – maksymalna wysokość obrazka
- ***input_file*** – pełna ścieżka dostępu do pliku graficznego, który chcemy przeskalować.

Podanie maksymalnej szerokości i wysokości obrazka pozwoli przeskalować pliki graficzne o zróżnicowanych proporcjach. Wiadomo, czasami zrobimy „sweetfocie” dłuższą w pionie, innym razem w poziomie. Ustawienie wartości granicznych pozwoli przeskalować obrazek tak, aby ani szerokość, ani wysokość przeskalowanego obrazka nie przekraczała ustalonych wartości maksymalnych. Przykładowo, jeśli mamy plik graficzny w rozdziałce 2560x1920, to ustawiając ***max_width*** i ***max_height*** na 1024 px, PIL przeskaluje dłuższą część obrazka do maksymalnej szerokości (1024 px) a wysokość obliczy tak, aby obrazek był nadal proporcjonalny. W efekcie otrzymamy obrazek o rozdzielczości 1024x768 px. Ale wróćmy do naszej funkcji. Jej nagłówek mógłby wyglądać następująco:

```
def resize_picture(max_width, max_height, input_file):
```

W ciele funkcji oddzielimy najpierw ziarno od plew, czyli z podanej ścieżki musimy wyłuskać ścieżkę do katalogu oraz nazwę pliku. Na przykład, jeżeli w zmiennej ***input_file*** mamy następującą ścieżkę dostępu do pliku graficznego: *„Foto:Wakacje2013/fotka_z_gor.jpg”*, to po rozbiciu jej następującymi operacjami:

```
dir = os.path.normpath(os.path.split(input_file)[0]);
file = os.path.normpath(os.path.split(input_file)[1]);
```

... w zmiennej ***dir*** zachowa się ścieżka do samego katalogu (czyli: *„Foto:Wakacje2013/”*), a w zmiennej ***file*** siedzieć będzie tylko nazwa pliku (a więc *„fotka_z_gor.jpg”*). To odseparowanie katalogu od nazwy pliku przyda się w czasie składania nowej nazwy dla przeskalowanego pliku (zdradzę, że ten nowy plik będzie miał prefiks *„resized_”*). Skalowanie odbywa się za pomocą wspomnianej już wcześniej metody ***thumbnail***. Funkcja ta przyjmuje dwa parametry: pierwszy określa nowy rozmiar obrazka, a drugi typ filtra:

```
picture.thumbnail((max_width,max_height),Image.ANTIALIAS)
```

W tym momencie obiekt *picture* przechowuje już nowy, przeskalowany obrazek. Jednak wciąż to nic „namacalnego”. Zapiszmy więc efekt naszej pracy na dysk. Aby odróżnić plik źródłowy od wynikowego, dodamy wspomniany już wcześniej prefix „resized_” do pierwotnej nazwy obrazka (którą przechowuje zmienna ***file***). Nową nazwę tworzymy w następujący

sposób:

```
new_filename = "resized_" + file
```

Przeskalowany plik zapiszemy w tym samym katalogu, w którym znajduje się plik źródłowy. Łączymy zatem katalog i nową nazwę pliku metodą ***join*** i jeszcze dla bezpieczeństwa normalizujemy całą ścieżkę funkcją ***normpath***.

```
picture_resized_path = os.path.normpath(os.path.join(dir,new_filename))
```

Jesteśmy gotowi do lotu. Odpalamy więc silniki oraz dobrze nam znaną metodę ***save*** z pełną ścieką dostępu do nowego pliku oraz stopniem kompresji jako parametrami.

```
picture.save(picture_resized_path,quality=85)
```

I tyle! Tak z grubsza wygląda metoda konwertująca pliki graficzne do zadanej rozdzielczości. Aby cały skrypt nabrał „mocy”, trzeba go jeszcze obudować główną metodą ***'__main__'***. Odczytamy w niej parametry ***max_width*** i ***max_height*** podane przez użytkownika oraz nazwę pliku. Oto jak powinna wyglądać:

```
if __name__ == '__main__':
if (len(sys.argv)>1):
max_width = int(sys.argv[1])
max_height = int(sys.argv[2])
for plik in sys.argv[3:]:
resize_picture(max_width, max_height, plik)
```

Główna funkcja naszego skryptu odczytuje najpierw argumenty z linii poleceń. Maksymalna szerokość ląduje w ***max_width*** a wysokość – wiadomo gdzie. Potem rozpoczyna się lot w pętli po wszystkich kolejnych argumentach podanych w linii poleceń. No właśnie i tu dochodzimy do sedna całego tego zamieszania. Tymi pozostałymi argumentami mogą być po prostu ścieki do plików, które chcemy skonwertować. Dla przykładu, chcemy przeskalować dwa pliki graficzne, aby ich szerokość lub wysokość nie przekraczała 1024 px. Uruchamiamy więc z poziomu CLI nasz skrypt z takimi oto argumentami:

```
python resize.py 1024 1024 ram:foto1.jpg ram:foto2.jpg
```

Skrypt wprawi w ruch „maszynę parową” PIL-a i już po chwili w RAM znajdziemy dwa nowe pliki: *resized_foto1.jpg* i *resized_foto2.jpg*. Fajnie, co?

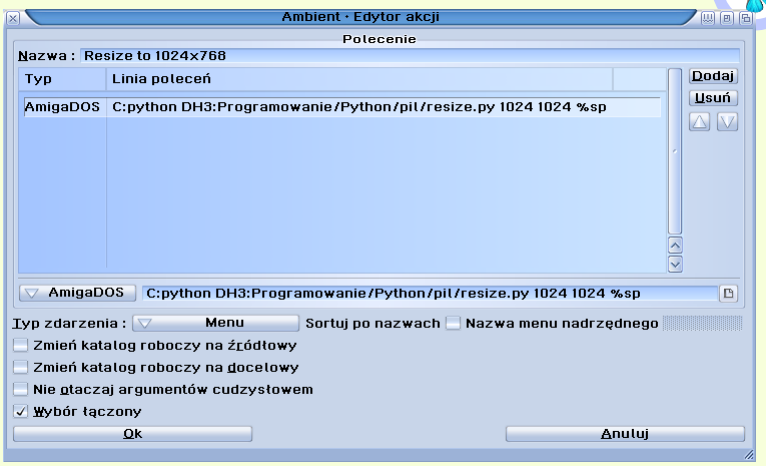
Deser

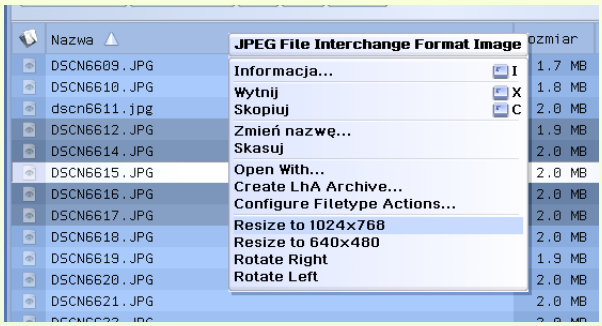
Wiemy już jak działa skrypt do hurtowej zmiany rozdzielczości obrazków napisany w oparciu o bibliotekę PIL. Jednak niewielu jest „artystów”, którzy za każdym razem mieliby ochotę odpalać okienko CLI i wklepywać tam całą litanię argumentów (wraz ze ścieżkami dostępu do plików graficznych). Dużo wygodniej byłoby, gdybyśmy mogli podpiąć nasz skrypt pod menu kontekstowe Ambienta (wywoływanego przez kliknięcie prawym przyciskiem myszy) i wybierać z listera pliki do skonwertowania. To właśnie zostawiłem Wam na deser.

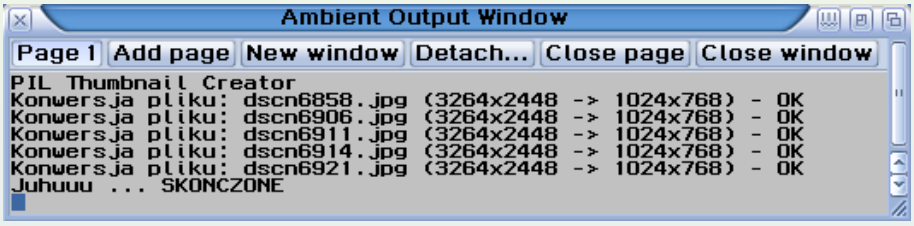
Przechodzimy do ustawień Ambienta i w zakładce MIME odnajdujemy definicję akcji dla plików typu *„jpeg”* (JPEG FileInterchange Format Image) w grupie *„image”*. Tworzymy nową akcję o nazwie *„Resize to 1024x768”*, a w linii poleceń umieszczamy wywołanie skryptu (pamiętajcie o podaniu pełnej ścieżki dostępu do skryptu):

```
C:python <ścieżka_do_katalogu_ze_skryptem>/resize.py 1024 1024 %sp
```

Typ zdarzenia to **Menu**. Koniecznie zaznaczamy opcję ***„Wybór łączony”***, aby skrypt uruchomił się tylko raz i odebrał jako argumenty kompletną listę ścieżek dostępu do naszych obrazków (w innym wypadku wykona się oddzielnie dla każdego z podanych plików). Zapisujemy ustawienia Ambienta i szukamy na naszym dysku jakichś biednych plików graficznych w formacie JPG, które mogłyby posłużyć nam jako króliki doświadczalne (spokojnie, nic im się nie stanie, najwyżej zostaną sklonowane). Zaznaczamy więc gromadkę obrazków i klikając prawym klawiszem myszy wywołujemy menu kontekstowe Ambienta. Tam powinien znajdować się już odnośnik do naszego skryptu pod nazwą *„Resize to 1024x768”*. Po jego wybraniu, wyświetli się jeszcze okienko konsoli, w którym na bieżąco będziemy mogli oglądać jak przebiega proces konwersji. Po dłuższej lub krótszej chwili PIL powinien wypluć skonwertowane pliki graficzne z prefiksem *„resized_”*.







Oczywiście nic nie stoi na przeszkodzie aby utworzyć drugą akcję w menu kontekstowym Ambienta i nazwać ją np. *„Resize to 640x480”* ze skryptem wywoływanym np. w następujący sposób:

```
C:python <ścieżka_do_katalogu_ze_skryptem>/resize.py 640 640 %sp
```

W ten sposób możemy skonwertować jednocześnie wiele plików graficznych do wielu różnych, predefiniowanych rozdzielczości. Nie musimy odpalać *ShowGirls*, wybierać żmudnie plików i za każdym razem ustalać docelowej rozdzielczości. Wszystko możemy mieć pod prawym przyciskiem myszy! Oczywiście nie żebym wieszał psy na *ShowGirls*, to wspaniały program, ale jego batch do konwersji obrazków zupełnie mnie nie przekonuje.

Podsumowanie

Tak przeczytałem ten artykuł i myślę, co by tu napisać na zakończenie. Właściwie może aż nadto rozpisałem się na temat tak prostych metod, jak te oferowane przez bibliotekę PIL, jednak wiem też, że niezwykle często diabeł tkwi w szczegółach i warto wiedzieć, co z czym się je i w jakiej kolejności. Tym bardziej nie chciałem wam podrzucać kompletnych skryptów na zasadzie czarnych skrzynek (tzn. podepnij pod Ambienta i hulaj dusza). Jeśli chodzi o sam proces konwersji za pomocą PIL-a, to muszę przyznać, że skrypt zżera bardzo dotkliwie czas procesora. Nie powiem, żeby konwersja odbywała się jakoś wyjątkowo powoli, jednak czuć, że to dość ciężkie zadanie dla procesora. Z pewnością konwersja w *ShowGirls* jest nieco szybsza, jednak biorąc pod uwagę fakt, że nasz pythonowy skrypt odpalamy jednym (a dokładnie to dwoma) kliknięciami myszy, można z całą pewnością uznać, że gra jest warta świeczki. Samo podpięcie skryptu pod menu kontekstowe Ambieta sprawia, że konwersja np. 30 zdjęć (które chcemy przesłać znajomym na maila) staje się banalnie prosta w realizacji. A myślę, że właśnie o taką wygodę tu przecież chodzi. Zresztą – sprawdźcie sami.

Archiwum ze skryptem znaleźć można na mojej stronie. Poza plikiem ***resize.py*** znajduje się tam pewien drobny bonus, ale to już nagroda dla wytrwałych lub ciekawskich.

Marek „MarX” Hać
fuego@o2.pl

Jeśli nie kręci was roztrząsanie poleceń Pythona na części pierwsze (czyli jakieś 80% tego artykułu), a z drugiej strony chcielibyście wzbogacić swój system o ten przydatny skrypt do konwersji zdjęć, to dobrze trafiliście. Wystarczy że zainstalujecie Pythona i bibliotekę PIL (opis na początku artykułu) i bez zbędnych ceregieli podepniecie skrypt ***resize.py*** pod menu kontekstowe Ambienta (zgodnie ze wskazówkami na końcu tego artykułu). Tak naprawdę nie potrzebujecie żadnej wiedzy nt. programowania ani tym bardziej na temat programowania w Pythonie. Enjoy!

Strona autora artykułu:
http://python.fatmagnus.ppa.pl/pil/

Strona, skąd można pobrać dodatek PIL:
http://effbot.org/imagingbook/

RISE OF THE TRIAD

Pod koniec roku 2013, po 11 długich latach od upublicznienia kodu źródłowego gry, dla Amig klasycznych ukazał się port pecetowej strzelaniny FPP – *„Rise Of The Triad: Dark War”* (w skrócie ROTT). Autorem portu jest pochodzący z Kansas koder, ukrywający się pod pseudonimem Lantus360. Niniejszy opis oparty jest na wersji 1.0 portu gry, jednak Lantus360 w wątku na forum EAB zapowiadał, że pojawią się kolejne wersje.

Gra *„Rise Of The Triad”* wydana została na komputery PC w roku 1994 przez firmę Apogee Software. W pierwotnym zamyśle autorów miała to być kontynuacja hitu *„Wolfenstein”*, jednak ostatecznie zrezygnowano z tego pomysłu. Fabuła *„ROTT”*, jak to w większości gier bywa, zbudowana jest wokół motywu walki dobra ze złem. Gracz wciela się w postać agenta organizacji H.U.N.T., a jego zadaniem jest zlikwidowanie członków niebezpiecznego kultu, który zajął jedną z wysp u wybrzeży Kalifornii. Ciekawostką jest fakt, że sekciarze ubierają się w mundury do złudzenia przypominające te, noszone przez hitlerowców. Interesującą cechą *„ROTT”* jest możliwość wyboru pomiędzy pięcioma postaciami, w które możemy się wcielić. Różnice między bohaterami są bardzo istotne – różnią się szybkością, celnością i wytrzymałością, więc początkowy wybór ma duże znaczenie podczas rozgrywki. Uzbrojenie naszego bohatera jest bardzo podobne do tego dostępnego w grach *„Wolfenstein”* i *„Spear of Destiny”*, czyli znajdziemy m. in. pistolet Walter i karabin maszynowy MP40.

Zasady gry są proste, należy zabijać wrogów, zbierać kolejne bronie i klucze, odnajdować przejścia. Każdy, kto grał w *„Dooma”* i *„Wolfensteina”* będzie się czuć jak w domu. Etapy gry są dosyć rozległe, już druga mapa zabiera kilkanaście minut. Po naciśnięciu klawisza TAB możemy obejrzeć mapę danego etapu, widoczni są też najbliżsi wrogowie oraz apteczki (w *„ROTT”* są one w postaci posiłków), bronie i klucze. Rozgrywka jest dynamiczna i krwawa, wrogowie padają na ziemię, prosząc o litość, po to tylko, żeby potem wstać i strzelić bohaterowi gry w plecy. Na zakończenie każdego z czterech epizodów gry, czeka nas walka z bossem, wielokrotnie natkniemy się też na pułapki, np. w postaci miotaczy ognia czy wirujących ostrzy, do których nie należy się zbliżać.

Technicznie engine *„ROTT”* plasuje się pomiędzy grami *„Wolfenstein”* i *„Doom”*, stopniem zaawansowania przypominając amigową strzelaninę *„Nemac IV”*. Mamy tutaj pełne teksturowanie podłóg i sufitów oraz zróżnicowaną wysokość pomieszczeń. Brakuje co prawda schodów, jednak jest możliwość wchodzenia na specjalne platformy i windy.

Wymagania amigowej wersji portu *„ROTT”* to system operacyjny w wersji co najmniej 3.1, procesor 68020, kości graficzne działające co najmniej w trybie EHB lub karta graficzna, 16 MB pamięci Fast i 20 MB miejsca na dysku. Oczywiście wymagane jest posiadanie pełnej, pecetowej wersji gry, z której należy skopiować pliki z danymi. Dystrybucją w wersji elektronicznej zajmuje się serwis GOG.COM, a cena tytułu to ok. 3 USD. Po zdobyciu odpowiedniej wersji czeka nas jeszcze jedna czynność, którą należy wykonać na pececie – chodzi o spatchowanie plików tak, aby amigowy port poprawnie odtwarzał dźwięk. Wszystko, co niezbędne znajduje się w archiwum z grą, a cała procedura opisana jest w dokumentacji.

Gra wymaga procesora 68020, jednak autor portu lojalnie informuje, że jest to za słaby procesor, aby czerpać z niej przyjemność – w praktyce rozgrywka staje się wygodna na 68040, a przy moim 68060/75 MHz uzyskuję ponad 20 fps przy największym oknie i pełnych detalach (są to wyniki dla chipsetu AGA). Niestety nie udało mi się uruchomić gry w trybie CGX – posiadam zainstalowane sterowniki P96 i kartę Voodoo III w Mediatorze i mimo dostępnego wymaganego przez grę trybu 320x200x8bit, gra nie uruchamia się. Według informacji od forumowiczów z English Amiga Board gra działa bez problemów, ale np. na karcie Cybervision 64, działającej pod kontrolą P96. Niestety nie uruchomimy *„Rise Of The Triad”* w rozdzielczości wyższej niż 320x200. Trochę szkoda, ponieważ myślę, że na 68060 i karcie graficznej z powodzeniem można byłoby zagrać w wyższej rozdziałce.

Tak jak wspomniałem na wstępie, autor zakłada dalszy rozwój portu. Zamierza m. in. zoptymalizować kod pod kątem szybkości działania na 68030, dodać obsługę joysticka oraz obsługę dla trybu Indivision ECS. Czas pokaże czy Lantus360 będzie miał motywację, żeby rozwijać port dalej.

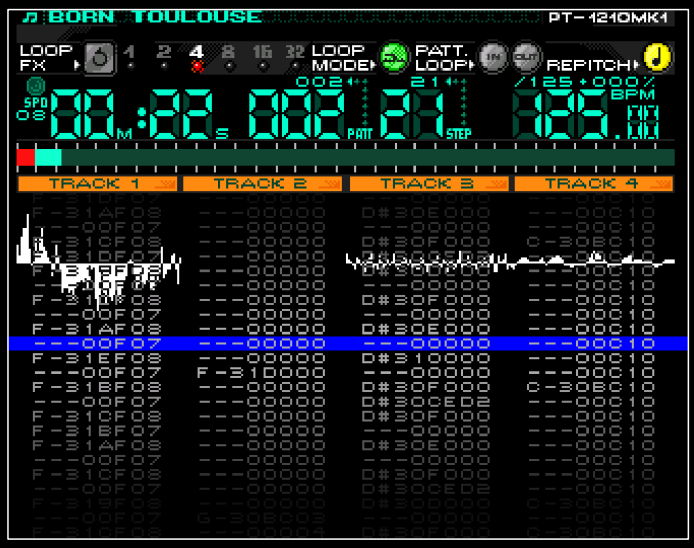
Czy warto zagrać w *„Rise Of The Triad”* na Amidze? Na pewno tak. Port jest stabilny i dopracowany, a rozgrywka prosta, ekscytująca i pełna miodności. Oczywiście, *„ROTT”* daleko jest do współczesnych FPP czy nawet *„Quake”*, jednak każdy, kto ma ochotę dla relaksu postrzelać i pobiegać po korytarzach gry, będzie usatysfakcjonowany.

Mateusz „Twardy” Eckert

Wśród amigowej braci znajdzie się niejeden DJ. I nie jeden pewnie myślał o tym, aby na imprezie zaprezentować muzykę prosto z ukochanej Amigi. Zagrać tak, jak gra się z profesjonalnego djskiego sprzętu – czy to z gramofonów, czy cdplayerów. Do niedawna jedynym sposobem było zgranie amigowego moda do pliku w formacie WAVE i wypalenie go na płycie CD. Te czasy się już jednak skończyły, bo nareszcie powstał program, który pozwala odtwarzać amigowe mody tak, jakby były odtwarzane za pomocą profesjonalnego, djskiego CD playera.

Czymże specjalnym jest taki sposób odtwarzania? Odtwarzacze dla DJ-ów pozwalają na regulację tempa utworu, zapętlanie, szybki powrót do zapamiętanego punktu i wiele innych ciekawych funkcji, które umożliwiają płynne miksowanie utworów muzycznych. To samo daje nam program PT-1210 MK1. Oferuje regulację tempa moda, zapętlanie patternów czy też szybkie zapętlenie wybranej ilości taktów. Dodatkową przewagą jest konstrukcja moda. Cztery ścieżki, które można dowolnie wyciszać dają wiele możliwości do eksperymentowania.

A teraz wyobraźcie sobie dwie Amigi, na których pracuje ten djski software i obie są wpięte w djski mikser audio – nieograniczone możliwości miksowania modów z dwóch komputerów. Już nie cztery a osiem ścieżek do zabawy. Z dwóch dowolnych modów, sprawnie operując tempem i ścieżkami, tworzy się zupełnie nowy utwór. Program ten daje możliwość, jakiej do tej pory nie było – miksowanie Protrackerowych modów na żywo. Oczywiście można iść dalej za ciosem i podpiąć do zestawu wiele innych urządzeń, takich jak gramofony, syntezatory, efektory. Ogranicza nas już tutaj tylko nasza kreatywna wyobraźnia.



Program, jak piszą autorzy Akira i Hoffman, napisany jest tak, aby bezproblemowo pracował na dowolnej klasycznej Amidze. Wystarczy umieścić swoje ulubione mody w katalogu, gdzie znajduje sie plik startowy i jesteśmy gotowi do pracy. Jedynym poważniejszym minusem, jaki został wychwycony podczas wesołych testów na Amiparty XV to fakt, iż do ładowania modów używana jest tylko pamięć Chip i niestety przy większych wagowo modach bywa, iż tej pamięci brakuje. Dodatkowym smaczkiem na stronie programu są paczki z selekcją modów od Akiry i Hoffmana. Sterowanie programem odbywa się głównie przy pomocy klawiszy funkcyjnych. Po całość klawiszologii, jak i dokładne specyfikacje odsyłam do pliku „readme" dołączonego do programu.

Podsumowując nie zawaham się użyć stwierdzenia, że na obecne czasy, jeżeli chodzi o wykorzystanie Amigi klasycznej, tego typu program jest niesamowitym krokiem naprzód. Otwiera szeroko bramę do występów na żywo, do prezentacji Protrackerowej muzyki na dużej scenie i to w sposób bardzo profesjonalny. Gorąco polecam..

Mariusz „MarioDJ" Jaczyński

Autor zdjęć z AmiParty XV – Stony

Odnośniki

Strona programu:
http://pt1210.kikencorp.com/

Zwiastun autorów oprogramowania:
http://vimeo.com/92446931

Moje boje i ich zapis audio:
https://soundcloud.com/mario_dj/amiga-mix-1-amiparty-xv

Video z Amiparty XV:
https://www.youtube.com/watch?v=pqCiPst6dIE

Goatlizard DX

Gra *„Moebius Goatlizard”* była już przeze mnie opisywana w 11. numerze PPA. Pojawiła się jednak dobra okazja, żeby do tego tytułu wrócić, bowiem grupa Blackjet postanowiła raz jeszcze uraczyć amigowców, wydając nową wersję tej gry. Szczegółowy opis rozgrywki zamieszczony jest w numerze 11, więc wystarczy przypomnienie, że *„Moebius Goatlizard”* to gra zręcznościowa, której celem jest zebranie wszystkich artefaktów na danym poziomie, przy jednoczesnym unikaniu kontaktu z przeciwnikami. Tytułowy Moebius może przemieszczać się po poziomych platformach za pomocą teleportów. W grze dostępne są także rozmaite power-upy, pomagające nam w osiągnięciu celu.

Nowa wersja gry nazywa się *„Goatlizard DX”*. Wprowadza sporo innowacji, dzięki którym rozgrywka staje się bardziej urozmaicona. Pierwszą, zauważalną zmianą jest dodanie muzyki – moduły odgrywane są zarówno na ekranie tytułowym, jak i podczas gry (razem z efektami dźwiękowymi) oraz po jej zakończeniu. Kolejną zmianą jest dodanie dużych, trudnych do uniknięcia bossów, zajmujących aż trzy sąsiadujące platformy. Pojawienie się bossów okazało się prostym zabiegiem, który w znacznym stopniu urozmaicił grę. Kolejne usprawnienia to poprawienie trybu dla dwóch graczy oraz wprowadzenie możliwości tworzenia własnych poziomów. Dzięki tej opcji, nazwanej Custom, możemy wybrać jedną z kilku dostępnych propozycji układów teleportów oraz zdecydować o typach przeciwników, którzy będą biegać po platformach.

Dodanie nowych opcji wpłynęło na wymagania sprzętowe gry, która nie uruchomi się już na Amigach wyposażonych jedynie w 512 kB pamięci (wymaga 1 MB pamięci Chip), a więc niestety nie pogram w nią na mojej ulubionej Amidze 1000. Dodatkowo autorzy informują, że gra nie działa na procesorze 68010 (który na szczęście w Amigach jest rzadkością), ale uruchomi się na procesorze 68000 i kicku 1.3.

http://blackjet.co.uk/amiga.php

Podsumowując, można stwierdzić, że wprowadzone zmiany wyszły grze na dobre. Nowy typ przeciwników oraz możliwości tworzenia własnych etapów znacznie urozmaiciły rozgrywkę, a dzięki dodaniu muzyki gracz ma wrażenie obcowania z kompletną, profesjonalną produkcją. Grę można pobrać za darmo ze strony autorów.

Korzystając z okazji, jaką jest wydanie *„Goatlizard DX”*, postanowiłem przeprowadzić wywiad z osobą odpowiedzialną za kod produkcji, Andrew Gillenem, znanym na forum EAB pod ksywą Spud.

Mateusz Eckert: Cześć, na wstępie powiedz proszę parę słów o sobie.

Andrew Gillen: Mam 37 lat, mieszkam w Oxfordshire, w Wielkiej Brytanii, pracuję w dziale IT, w firmie z sektora finansowego w Londynie. Moje główne Amigi to A1200 z kartą Blizzard 1230 IV oraz Amiga 4000 z Apollo 4040. Niestety nie mam tyle miejsca, żeby moje Amigi były cały czas podłączone i gotowe do działania, więc najczęściej pracuję na WinUAE. W swojej kolekcji posiadam także CDTV, CD32, A600, martwą A500+, A4000 przełożoną z obudowy desktop do wieży oraz A3000.

M.E.: Jak zaczęła się Twoja przygoda z Amigą i programowaniem?

A.G.: Pierwszą Amigę dostałem na urodziny, w roku 1991. Była to A500 w zestawie Screen Gems – zakochałem się w niej od pierwszego wejrzenia. Do niedawna nie zajmowałem się programowaniem, byłem tylko zapalonym amigowym graczem. Pomimo posiadania komputerów PC i różnych konsol, cały czas grałem na Amidze i bawiłem się takimi programami, jak Vista czy Deluxe Paint.

M.E.: Kim jest Moebius Goatlizard?

A.G.: Cóż, Moebius jest postacią wykreowaną przez Davida Hughes na potrzeby jego gier na ZX Spectrum, takich jak *„Stamp Quest”*, *„Endless Forms Most Beautiful”* i *„The Lost Tapes of Albion”*. Moebius jest dziwacznym i jednocześnie urokliwym spritem. Co do jego nazwiska, Goatlizard, to wymyślił je Jaco (grafik grupy), na etapie ustalania tytułu gry.

M.E.: Jak wpadłeś na pomysł stworzenia gry?

A.G.: Tak jak powiedziałem wcześniej, koncepcję gry stworzył David Hughes. Uwielbiam jego gry na ZX Spectrum i najpierw postanowiłem napisać moją własną wersję *„Lost Tapes of Albion”* na komputer Sam Coupe, ponieważ uznałem go za odpowiednią platformę do przećwiczenia swoich umiejętności i upiększenia gry w stosunku do oryginału. Przez ostatnie parę lat moje zainteresowanie retro grami spowodowało niechęć do grania w cokolwiek „nowoczesnego”, zapragnąłem także zgłębić swoją wiedzę na temat architektury retro komputerów. Naukę zacząłem od asemblera Z80 na ZX Spectrum i Sam Coupe, a potem zainteresowałem się Amigą i asemblerem 68k. Jako, że posiadałem już gotową grę dla Sama Coupe, stwierdziłem, że przeniesienie jej na Amigę nie będzie stanowiło problemu i że projekt jest możliwy do zrealizowania.

M.E.: Z kim współpracowałeś przy tworzeniu gry?

A.G.: Pracowałem z grafikiem, Jaco van der Walt-em. Ja sam potrafię ewentualnie przerobić czyjąś grafikę z gry na inną platformę (takie, jak np. sprite'y Davida Hughesa), nie jestem jednak na tyle dobrym grafikiem, żeby stworzyć coś od podstaw. Jaco tworzy dużo grafiki, zajmuje się także testowaniem oraz tworzeniem efektów dźwiękowych w projektach Black Jet.

M.E.: Czy jesteś bądź byłeś kiedyś związany z amigową demosceną?

A.G.: Nie. Do niedawna nawet się tym nie interesowałem. Na chwilę obecną, od kiedy lepiej rozumiem, co kryje w sobie Amiga, zacząłem bardziej doceniać produkcje demosceny, jednak to Jaco jest bardziej zainteresowany tematami scenowymi.

M.E.: Czy planujecie wydanie kolejnych gier na Amigę? Co się stało z zapowiadaną grą *„Carpet Capers”*? Pamiętam, że na waszej stronie były już zapowiedzi i nawet trochę grafiki z gry.

A.G.: Gra *„Carpet Capers”* nie była zaawansowana w rozwoju. Początkowo była to gra, którą pisałem na Sama Coupe, chcąc jednocześnie nauczyć się więcej o programowaniu scrollowania. Pomyślałem, że fajnie byłoby pisać jednocześnie na Amidze, jednak bardzo szybko poczułem się wypalony. Tworzyłem jednocześnie gry na Amigę, ZX Spectrum i Sam Coupe, urodził mi się także syn, więc miałem mniej czasu. Postanowiłem zrezygnować z niektórych pomysłów i skupić się na prostszych tematach, takich jak strzelanina na ZX Spectrum oraz kolejna wersja Moebiusa na 25-lecie komputera Sam Coupe. Postanowiłem też wydać poprawioną wersję gry na Amigę, dodając muzykę i parę ciekawych dodatków.

Od tego zestawu zaczęła się przygoda autora gry z Amigą.

„Carpet Capers” miało być grą w stylu *„Space Taxi”*, ale osadzoną w arabskich klimatach. Może kiedyś do niej wrócę, podobała mi się grafika i zapowiadała się niezła frajda z programowaniem gry. Myślę jednak, że powstanie tylko wersja na Sam Coupe.

Co do nowych projektów związanych z Amigą, to na pewno coś planujemy. Nie będzie to jednak kolejna wersja Moebiusa – szczerze mówiąc mam go już dość! Chcemy stworzyć coś zupełnie nowego i jednocześnie chcę nauczyć się programowania kości AGA, więc gra będzie najprawdopodobniej na Amigę 1200. Pisanie gry to długi proces, a kiedy ma się mało czasu, proces ten ulega dalszemu wydłużeniu... Ech, fajnie byłoby programować na Amidze na pełen etat! W każdym razie, mamy już pewne notatki, a Jaco bardzo chce stworzyć amigowy odpowiednik gry *„Spacewhale”*, więc poczekajmy i zobaczmy, co się wydarzy.

M.E.: Jakie są twoje ulubione amigowe gry?

A.G.: *„Rainbow Islands”*, *„SWOS”*, *„Pinball Dreams”*, *„Monkey Island”*, *„Super Twintris”* oraz *„The Adventures of Robin Hood”*. Oczywiście jest także wiele innych, jednak te które wymieniłem, przyszły mi do głowy jako pierwsze, więc pewnie to właśnie one są moimi ulubionymi.

M.E.: Na zakończenie powiedz proszę, czy interesujesz się nowymi amigowymi i postamigowymi systemami, takimi jak AmigaOS 4, MoprhOS czy AROS?

A.G.: Nie interesuję się tym. Wypróbowałem kiedyś AROS-a, ale ponieważ na co dzień pracuję w branży IT, to po godzinach pracy nie interesują mnie żadne nowoczesne rzeczy, niezależnie czy mówimy o PC, nowoczesnych konsolach, czy nowych amigowych systemach. Relaksuję się obcując ze starymi maszynami z czasów młodości.

M.E.: Dziękuję za rozmowę.

Mateusz „twardy” Eckert

STAR WARS™ JEDI KNIGHT™: JEDI ACADEMY™

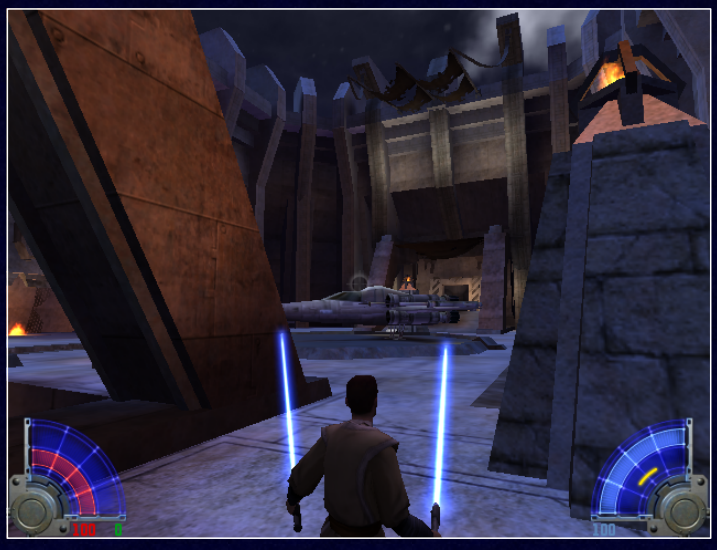
Dawno temu, w odległej galaktyce żył sobie pewien chłopiec. Miał wiele przygód – poleciał ratować księżniczkę, rozwalił orbitalną stację bojową, stał się rycerzem Jedi, odnalazł ojca, stracił dłoń, odzyskał dłoń, potem pokonał złego Imperatora, stracił ojca, ale za to odnalazł siostrę. W pojedynkę – no, może samotrzeć – zadał druzgocącą klęskę złemu Galaktycznemu Imperium i było wiele radości. Ale z takiej historii można ciągnąć kasę niemal w nieskończoność, więc nie myśleliście chyba że na tym się skończy?

11 lat po pierwszej bitwie o Yavin – Imperium, mimo otrzymania druzgocących ciosów nadal istnieje i wciąż próbuje kontratakować, choć już raz kontratakowało i nic dobrego z tego nie wyszło. Ponieważ – jak się okazuje – Luke Skywalker jest dość leniwy i nie ma już ochoty na samotną walkę z Siłami Zła, założył na Yavin 4 Akademię Jedi, do której ściąga młode talenty z całej galaktyki. Jednym (lub jedną – zależy co wybierzecie w opcjach) z nich jest Jaden Korr, który (lub która) właśnie próbuje się zrelaksować na pokładzie promu, mknącego na powierzchnię planety niczym nieubłagane ostrze losu.

Jak już pewnie zdążyliście się przyzwyczaić w przypadku wielu innych pojawiających się ostatnio gier na nasze amiplatformy, rozgrywkę w *Jedi Academy* rozpoczynamy od dwóch rzeczy: pobrania portu oraz zdobycia wersji pecetowej. Następnie przegrywamy wszystkie potrzebne pliki do podkatalogu */base* i gotowe! Prawda, że dużo prostsze od uruchamiania jakichś instalatorów?

Fabuła gry – jak pewnie się domyślacie – nie będzie polegać na słuchaniu wykładów na temat „Jak skutecznie nie dać się przeciągnąć na ciemną stronę mocy" albo „Jak zachowywać się w kantynie na Zewnętrznych Rubieżach, żeby nie spowodować intergalaktycznego konfliktu". Takie rzeczy to może były lata temu, gdy młodzików Jedi szkolił ten wapniak Yoda. Dziś uczymy się na sposób praktyczny. Krótki kurs jak machać mieczem świetlnym jak najdalej od krocza swojego mistrza (którym nota bene jest znany i lubiany jeszcze od czasów *Dark Forces* czy *Rebel Assault* – Kyle Katarn) i jesteś gotów (lub gotowa – zależy co wybierzesz w opcjach) do misji! A tych jest trochę do wykonania, bo cień jakowyś padł na Galaktykę – coraz częściej słyszy się o jakichś Uczniach Ragnosa, że niby zbierają gdzie się da końcówki Mocy i pewnie chcą jej użyć w jakimś niecnym celu. A jeśli cel jest niecny, to kto wkracza? My! I kogo będą pokazywać na holo stąd do Ord Mantell? Nas!

Pora na kilka słów o oprawie audiowizualnej: Po pierwsze – podstawą gry jest (podobno niebywale ulepszony, ale jakoś tego nie zauważyłem) silnik *Quake 3*, co powinno dać Wam już jakiś osąd. Jeśli nie daje, to powiem tak: jak na amiwarunki jest całkiem nieźle, choć okupione jest to podwyższonymi wymaganiami sprzętowymi – Mac Mini wyposażony w Radeona 9200 z 64 MB VRAM lubi niestety dostać zadyszki (z typowymi dla braku pamięci graficznej artefaktami na ekranie), więc niestety trzeba będzie pogrzebać w opcjach i poświęcić rozdzielczość, tudzież detale na ołtarzu płynności. Po drugie – BSzili robił co mógł, ale z racji że TinyGL mamy taki, a nie inny w niektórych miejscach spotkamy się z uproszczeniami wyświetlania, których nie uświadczymy w wersji „pecetowej" (na szczęście nie ma ich dużo). Po trzecie – lokacje są zaprojektowane z elegancją, niektóre nawet z rozmachem, a udźwiękowienie i muzyka trzymają klimat (kto nigdy nie stał przed lustrem wymachując świetlówką mrucząc *„łu... łu..."* ten i tak nie zrozumie), a twórcy gry zadbali nawet o takie detale, jak skwierczące wesoło i parujące na ostrzu naszego miecza świetlnego kropelki padającego deszczu.



Skoro o mieczu mowa – mówiłem Wam już, że to elegancka broń, na bardziej cywilizowane cza... Nie, nie – chodziło mi o to, że w *Jedi Academy* solidnie się nim namachamy (i to w trybie TPP). W dodatku jest to jedyna broń, do której nie potrzebujemy amunicji. Dla uparciuchów – można przejść całą grę korzystając tylko z niego (za wyjątkiem jednej misji), choć do dyspozycji mamy wiele blasterów, a nawet wyrzutnię rakiet, że o termodetonatorach i minach-potykaczach nie wspomnę. Z kolei dla Hardkorów – można spróbować tylko trybu FPP, bez użycia miecza świetlnego (rozgrywka skończy się bardzo szybko, gwarantuję). Jest oczywiście też łyżka dziegciu w całej tej świetlnomieczowej awanturze – nasz sprzęt jest najzwyczajniej w świecie wybrakowany! To nie są te same ostrza, które topią zbrojony metal czy wycinają dziury w burtach kosmicznych krążowników! Te błyskotki może i potrafią rozczłonkować jakiegoś Mrocznego Jedi (ale nie tak od razu – najpierw z reguły trzeba go mocno posiekać), zwalić drzewo czy przeciąć łańcuch (z tym, że nie każde, tylko konkretne, wybrane przez twórców gry), ale próba głupiego wycięcia zamka z zablokowanych drzwi co najwyżej zakończy się pozostawieniem na nich nikłego, wypalonego śladu, który zaraz zniknie. Co to ma być? Potężna broń Jedi czy jakieś – nie przymierzając – żelazko Philips Elance

- dopracowana oprawa audiowizualna,
- zróżnicowane misje,
- gratka dla fanów Gwiezdnych Wojen.

- liniowość rozgrywki,
- pewne niedociągnięcia/niekonsekwencje w grze

3100? Zostaliśmy bezczelnie oszukani i żądamy zwrotu pieniędzy, ot co! A raczej żądalibyśmy, gdyby nie to, że pojedynki na miecze świetlne to – mimo wszystko – wielka siła tej gry, szczególnie jeśli władamy naszą bronią biegle (a z czasem dostaniemy do wyboru trzy style walki, a nawet wymianę naszego wysłużonego miecza na dwa albo miecz podwójny). A jeśli dołożymy jeszcze techniki Mocy...

A właśnie – Moc. To kolejna sprawa, o której nie można nie wspomnieć. Oprócz podstawowych technik Mocy, jak przyciąganie, pchnięcie, szybkość czy skok, nasz bohater (lub bohaterka – zależy co wybierzesz w opcjach) ma do wyboru osiem mocy specjalnych (po cztery dla jasnej i ciemnej strony). To od nas zależy które z nich chcemy rozwijać w pierwszej kolejności – osobiście polecam leczenie (pozwala uniezależnić się od rozsianych po planszach apteczek – niestety nie ma umiejętności Mocy regenerującej utraconą zbroję czy amunicję) oraz duszenie (na wyższych poziomach pozwala w zabawny sposób pozbyć się adwersarza – na przykład przenosząc go nad jezioro lawy i zwalniając chwyt). Niestety – za wszystko trzeba płacić, poziom mocy spada przy każdym użyciu, a regeneruje się dość wolno. Warto więc śledzić wskaźnik po prawej, szczególnie jeśli akurat zachciało się nam wykonać Skok Mocy nad pierwszą-lepszą, bezdenną otchłanią. Aha, nie przejmujcie się, że używając umiejętności Mocy spod znaku Ciemnej Strony staniecie się ni stąd i zowąd

Darthem Korrem – wybór co prawda dostaniemy, ale dopiero pod koniec gry (i od niego będzie zależeć ostatnia misja oraz finałowy pojedynek).

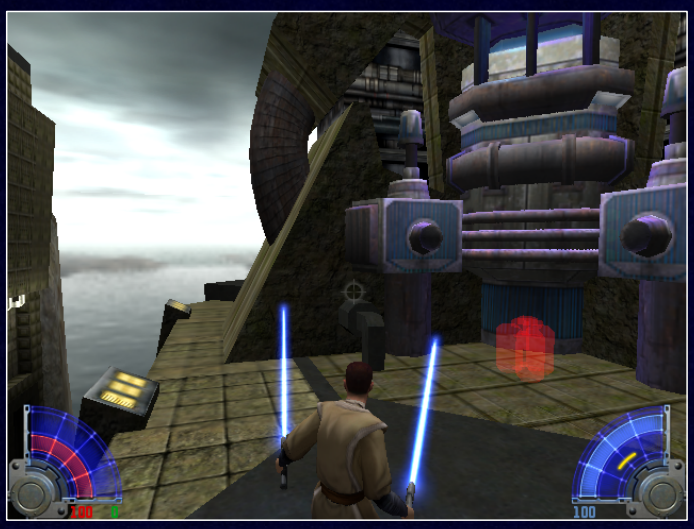
Ale ja tu już o ostatniej misji, a Wy jeszcze się nie nauczyliście, którą część miecza się trzyma, a którą macha. Ogólnie – misje są bardzo zróżnicowane. Przyjdzie nam podkładać ładunki wybuchowe, rozbrajać ładunki wybuchowe (rzecz jasna nie te same), ratować zakładników, szukać części do Maka na szrocie... (napisałem „do Maka?" Miałem na myśli: „do statku", oczywiście), aresztować złoczyńców, uciekać z więzienia, ba – nawet sabotować imperialny statek bojowy! Zwiedzimy przy tym pół galaktyki, nie omijając oczywiście standardowych lokalizacji typu Tatooine, Hoth, czy Coruscant. Misje danego rozdziału wykonujemy co prawda w dowolnej kolejności, ale – niestety – nie da się uciec od przemożnego wrażenia, że gra prowadzi nas cały czas „za rączkę" – poszczególne zadania są liniowe do bólu, a czynności wykonujemy (z drobnymi wyjątkami) zawsze w tej samej kolejności. Nawet jeśli uda się Wam gdzieś utknąć, to pewnie z opresji wybawi Was moc „Wyczuwania (Sense)", ujawniająca ukryte przyciski, wajchy i takie tam. Jeśli kombinujecie za bardzo, to mechanizmy rządzące grą z pewnością Was zadziwią – jako dobry przykład niech posłuży misja, w której ratujemy więźniów z przedziwnej gry bukmacherskiej, polegającej na obstawianiu jak długo przeżyją uciekając przed Rancorem. Poświęcając wystarczająco dużo czasu i amunicji (tudzież ciosów mieczem) jesteście w stanie bestię powalić. Cóż jednak z tego, skoro drań zaraz się zrespawnuje? To nie fair, panie Lucas, po prostu nie fair...

Mimo wszystkich niedociągnięć w *Jedi Academy* zagrać warto, nawet jeśli (jak ja) nie jesteście zagorzałymi fanami serii *Star Wars*. Rozczarują się pewnie ci, którzy liczyli na coś w rodzaju *Dark Forces* na sterydach – tutaj trzeba jednak trochę pomachać mieczykiem w trybie TPP, co dla nienawykłych może rodzić pewne problemy. No i trzeba będzie przełknąć gorzką pigułkę wymagań sprzętowych.

Konrad Czuba

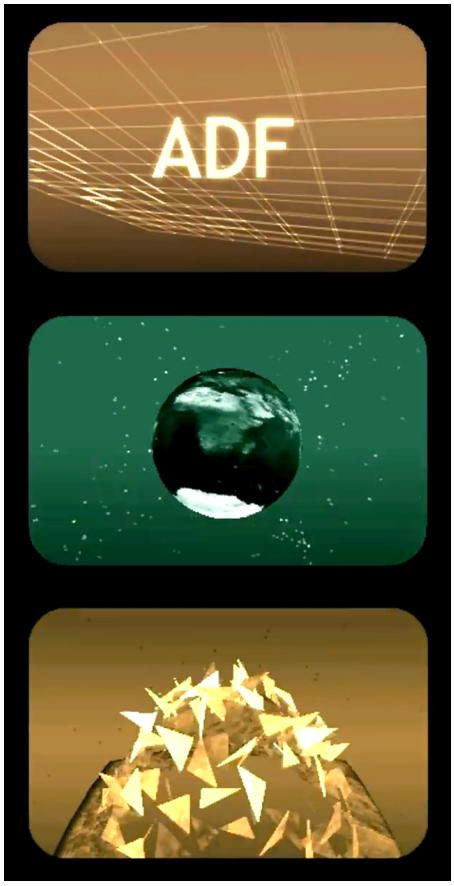
Projekt OpenJK:
http://jkhub.org/page/index.html

Strona autora portu:
http://bszili.morphos.me/



Scenowe graffiti

W poprzednim odcinku wspominałem na łamach PPA o nowej (w kontekście Amigi) grupie **Australian Drug Foundation**. Jak się okazało *„Feeling Blue”* nie było ich jednorazowym amigowym wybrykiem. Na party **Stream**, które odbyło się w dniach 16-18 maja w fińskim Tampere ADB zaprezentowali kolejną produkcję. *„Outside Planet Earth”* (kod – Shadez, muzyka – Serpent) znów ujmuje widza charakterystyczną melancholijną i rozmarzoną atmosferą, znaną z poprzedniego dema. Ładne, „świetliste” efekty, dobrze zsynchronizowane z muzyką a'la Jean-Michel Jarre tworzą przyjemną mieszankę. Co ważne, mimo oczywistej nostalgicznej aury, nie czuć tu owego „retro - geriatrycznego” wyrachowania, które to często owocuje produkcjami wydawanymi współcześnie, a wyglądającymi jak wyjęte żywcem z roku 1988 (w czym nigdy nie widziałem większego sensu). Gdyby tak jeszcze tylko do utalentowanego kodersko-muzycznego duetu Shadez + Serpent dołączył jakiś grafik z prawdziwego zdarzenia, grupa Australian Drug Foundation mogłaby wyrosnąć na nową gwiazdę amisceny, bo do pełni szczęścia brakuje mi tam już tylko solidnej oprawy graficznej.



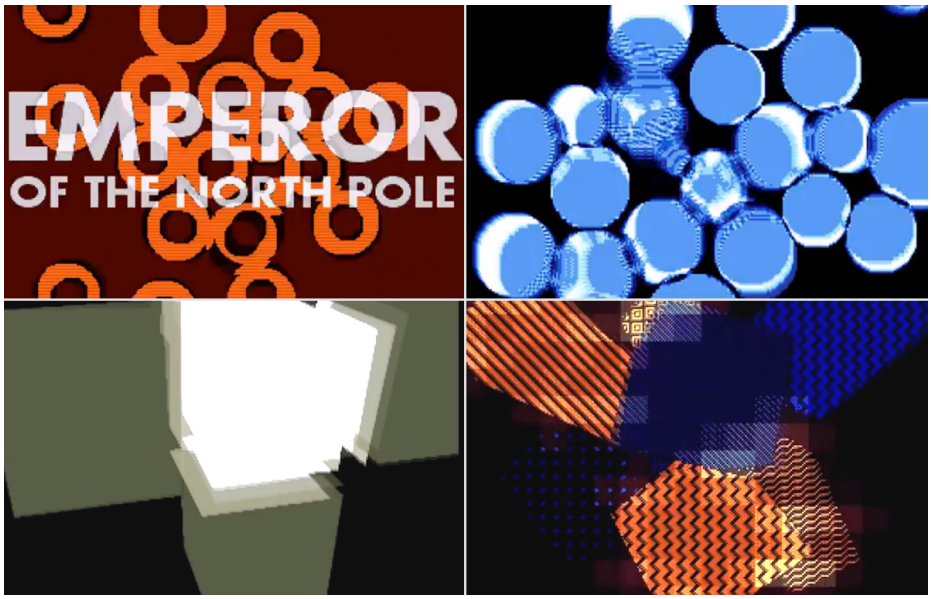
Demo „Outside Planet Earth” grupy Australian Drug Foundation

Solskogen to z kolei norweskie party, odbywające się (z małymi przerwami) od 2002 roku we Flateby – małej, malowniczej wiosce, położonej niedaleko Oslo. Jedyny polski wysłannik na tej imprezie, zakamuflowany agent rosyjskich służb specjalnych – Sachy - opisuje ją tak: *„Impreza jest na wsi, można spać w lesie w namiotach lub budynku wiejskiego domu kultury, w którym jest sala kinowo-dyskotekowa i tam też było głownie party. Ludzi około 200, Spaceballs, Insane, Hoaxers, Darklite, Szwecja, Finlandia, Norwegia, Niemcy, Szwajcarzy, ktoś z USA, UK no i ja sam z Polski. Atmosfera fajna, upał, więc party było w środku, a na zewnątrz piknik, grill i dmuchany basen z wodą itd. Klimat z jednej strony niby piknikowy, ale z drugiej naprawdę kawał dobrego party.”*

Demo „Sun Candy” grup Resistance i Wanted Team

Mimo, że w ostatnich latach **Solskogen** nie było jakoś specjalnie nasączone amigowymi produkcjami, tegoroczna edycja przyniosła w tej materii pewne pozytywne zmiany. Dość nieoczekiwanie to nie Newschool / PC Demo Compo, a Oldskool Demo Compo stało się głównym punktem imprezy. Wystawiono w nim 9 produkcji, z czego pięć było amigowych. Trzy z nich zajęły miejsca na podium i właśnie o tej zwycięskiej trójce warto coś więcej napisać. Demo *„Sun Candy”*, które zajęło miejsce trzecie, to kooperacja grup **Resistance** i **Wanted Team**. Jest to produkcja stworzona praktycznie w całości przez naszych rodaków i pierwsza – od bardzo dawna – większa rzecz, którą zakodował wspominany Sachy. Graficznie wspomógł go Sim1, a muzycznie Nainain i Mccnex. *„Sun Candy”* utrzymane jest w nieco oldskoolowym klimacie, przywodzącym (między innymi za sprawą muzyki) miłe, nostalgiczne wspomnienia. Design jest może momentami trochę niespójny, ale naprawdę świetne, ręcznie rysowane logosy autorstwa Sim1 odwracają uwagę od tego faktu. Jak powiedział cytowany wcześniej koder, pierwotnie miało to być tylko małe cracktro, ale wraz z upływem czasu zaczęły dochodzić coraz to nowe części, wydłużono przy okazji muzykę i zrobiło się dentro, a właściwe to małe demo. Kolorystycznie inspirację czerpali z dem znanych i lubianych grup Melon Design i Lemon, co również można zauważyć. Typowo „sceno-wy proces produkcji” dobrze podsumowuje wypowiedź Sachy'iego: *„Dentro skończyłem w noc przed wyjazdem (odlot 6 rano)”*.

Srebro w konkursie zdobyło demo na kości AGA i szybszy procesor. Mowa o *„We Are The Scene”* (kod – OriGO, Prospect, muzyka – Mygg, Gemini, grafika – Bitflippr, Corel, Randy), wspomnianej zresztą ostatnio przeze mnie na łamach *„Scenowego Graffiti”* – w kontekście powrotów na scenę po latach – grupy **Insane**. Z pewnością nie mamy tu zbyt zaawansowanych efektów, rażą też trochę „dłużyzny”, ale jest dobre synchro, niezły design i całkiem



Demo „Emperor of the North Pole” grupy Spaceballs

spójna całość. Widać, że Insane po swoim amigowym comebacku wzięli się solidnie do roboty i zaczynają powoli wypracowywać własny, ciekawy i unikatowy styl.

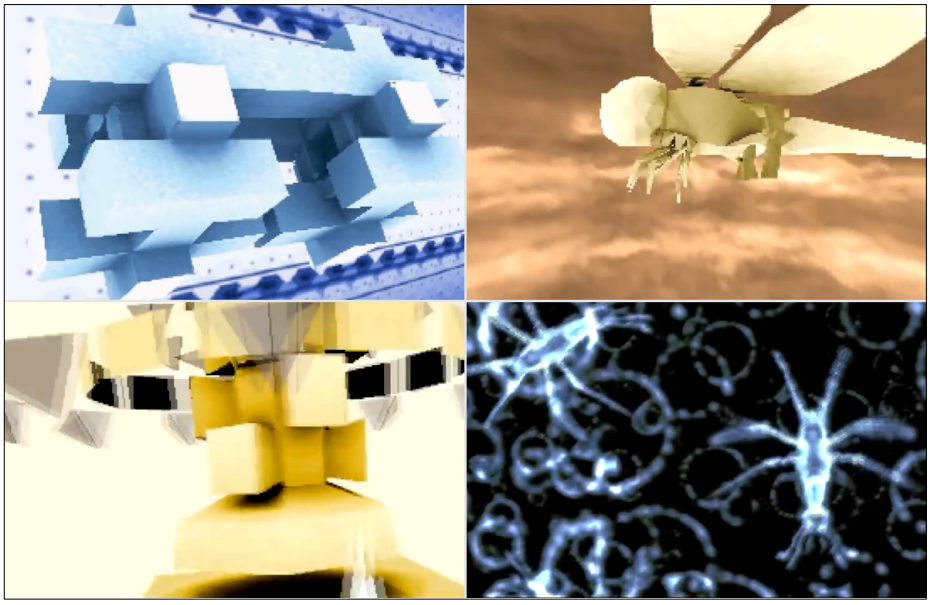
Czas na opis dema z miejsca pierwszego. Już sam fakt, że to produkcja na Amigę 500, która wygrała z demem na kości AGA może zaintrygować. *„Emperor of the North Pole”* grupy **Spaceballs** to naprawdę mocna pozycja. Slummy (odpowiedzialny za kod) to bez wątpienia jeden z ostatnich prawdziwych koderskich wizjonerów, który stara się tworzyć nowoczesne dema na „małe Amigi”, często wykorzystując przy tym koderskie, hardware'owe sztuczki, o których niewiele osób już dzisiaj pamięta. Jego produkcje nie przypominają tych, znanych z lat 90-tych. Są maksymalnie współczesne, kojarzące się bardziej z niektórymi minimalistycznymi demami na PC. *„Emperor of the North Pole”* to trzecia część jego „koderskiej trylogii” (poprzedzonej równie dobrymi demami *„Vold”* i *„Straff”*), która zawiera rozwinięcia i wariację wcześniej pokazanych efektów. Przyznam, że dawno nie widziałem tak dobrych procedur na układy OCS. Doskonała dynamika, świetne synchro pod świeżą, klubową ścieżkę, za którą stoi lug00ber, sprawia, że zastanawiamy się czy to oby na pewno demo na starą, poczciwą „pięćsetkę”, czy też może na większe Amigi. Po raz kolejny Slummy potwierdza, że Cesarz Bieguna Północnego jest tylko jeden.

Z wyjątkowo słonecznej Norwegii przenosimy się do – równie słonecznego – miasta Kraju Basków – Bilbao. To właśnie tam odbyło się demoscenowe party **Euskal**, na którym (można rzec, że tradycyjnie) dała o sobie znać amigowa grupa **Software Failure** z Hiszpanii. Tak naprawdę jest to projekt kierowany przez jednego człowieka, który bardzo sporadycznie zaprasza do gościnnej współpracy inne osoby (może to błąd). Ham, bo taką ksywę ma ów solista, to prawdziwy multiinstrumentalista – koder, muzyk i grafik w jednym ciele. Jak mówią plotki, lubi się on zaszyć w leśnej głuszy, z dala od cywilizacji i żywiąc się tylko tym, co sam upoluje, kodować dema. Może dlatego, w niemal wszystkich jego produkcjach napotkać możemy rozmaite insekty, owady i inne – mało smaczne nawet dla samego Roberta Makłowicza – stworzenia.

„Mind Floating Palace” (kod, grafika, muzyka – Ham) – demo na procesor 060 + AGA, które

Cover magazynu „Hugi #38 - Gravity Shot”, autor: Forcer

przegrało w łączonym Demo Compo jedynie z legendą demosceny, czyli grupą Andromeda Software Development i ich doskonałym demem na PC *„Mrs Escher's Nightmare”* – to tylko pozornie produkcja w typowej dla Software Failure estetyce. Moim zdaniem jest to spory krok do przodu w dokonaniach Hama. Z pewnością wciąż nie jest to poziom, jaki prezentuje TBL, Skarla czy Dekadence, ale w końcu więcej tu finezji i nacisku na design oraz klimat.



Demo „Mind Floating Palace” grupy Software Failure

Mamy zatem powolne, monumentalne obiekty 3D, nieco „organiczne”, laboratoryjne efekty i tajemniczy, spokojny soundtrack. Warto dodać, że na Euskal 2014 grupa SF wydała jeszcze amigowe 64k intro *„Flow My Programs”* – nie tak udane jak wspomniane demo, ale również prezentujące solidny poziom.

W czerwcu ukazał się 38 numer magazynu „Hugi”. Tematycznie traktuje on scenę przekrojowo i multiplatformowo, ale ortodoksyjni fani amisceny znajdą w nim między innymi tekst dotyczący prac nad dokumentem *„Viva Amiga”*, wywiad z Paradroidem / Lemon, Hoffmanem / Unstable Label, raport z Datastorm czy publikację o legendarnym diskmagu *„RAW”*. „Hugi” w kodowanej wersji dostępny jest niestety tylko dla systemu Windows, ale Magic / Nah-kolor (zastępca naczelnego) obiecał mi, że „niedługo” pojawi się także wersja dla systemu AmigaOS 4.0. Niezależnie od tego cały czas dostępna jest też wersja online.

Na koniec jeszcze krótka wzmianka o produkcjach wydanych na kilku mniejszych imprezach, które to z różnych powodów warto odnotować. Na niewielkim party **RESETkani**, które odbywa się cyklicznie w czeskim Brnie, dała o sobie znać (po bardzo długiej przerwie) amigowa formacja **D.T.A. Software Studio**. Ich 64k intro *„Reborn”* nie jest może niczym porywającym, ale znak życia z „wymarłych amisceonwo” sąsiednich Czech jest z pewnością godny uwagi. Na jeszcze mniejszym – tym razem polskim – meetingu **Garaż Party 2014** zaktywizowała się (w kontekście amigowowym) grupa **Lamers**. Ich *„Greetz-tro”* (kod, grafika – at0m, muzyka – Lamesoft) to – co prawda – króciutka, prosta, pchełka, ale ogląda się ją z naprawdę dużą przyjemnością. Nie wiem, czy to był tylko „amigowy wypadek przy pracy”, ale chętnie obejrzałbym w końcu jakieś nowe produkcję Lamersów na naszej ulubionej platformie, a nie na (nieco mniej ulubionym) Atari. :-)

Tomasz Pacyna (Slayer/Ghostown)
info@pixeldreams.pl