FBLITGUIDE

Stephen Brookes

COLLABORATORS

	TITLE : FBLITGUIDE	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Stephen Brookes	August 24, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FBLITGUIDE

1.1 FBlit

```
                FBlit v3.66


                   Ostrzeûenie

                   Dystrybucja

                   Co To Jest

                    Wymagania

                   Instalacja

                     Obsîuga

                    Problemy

                 Kto To Zrobiî?

                   Historia

                   FBlitGUI

                  Listy Zadaï

                   Nakîadki
```

1.2 Ostrzeûenie

```
 FBlit, FBlitGUI i fblit.library sâ © Stephen Brookes 1997 - 2000

 Oprogramowanie zawarte w tym archiwum jest nieskoïczone (beta),
z natury eksperymentalne i niebezpieczne, wiëc go nie uûywaj.
Nie biorë odpowiedzialnoôci za ûadne niepoûâdane efekty wynikajâce
```

z uûywania oprogramowania i informacji zawartych w tym archiwum.

1.3 Dystrybucja

To archiwum moûe byê swobodnie rozpowszechniane.

Najnowsza wersja FBlita jest dostëpna na...

<http://www.tpec.u-net.com>

Zanim poinformujesz mnie o bîëdzie w programie, upewnij sië ûe masz jego najnowszâ wersjë pobranâ z tego adresu.

1.4 Co To Jest

Co to jest?

FBlit jest hackiem, który powinien zredukowaê iloôê Chip RAM'u uûywanego przez aplikacje bazujâce na systemie, na Amigach ze zwykîymi ukîadami graficznymi (czyli bez kart graficznych). Moûe teû przyspieszyê niektóre operacje i zredukowaê migotanie kolorów.

Tak, ale co to jest?

FBlit jest zbiorem nakîadek, które umoûliwiajâ systemowi obsîugë danych graficznych znajdujâcych sië poza pamiëciâ Chip.

Normalnie nie jest to moûliwe, poniewaû systemowe funkcje przetwarzajâce grafikë uûywajâ ukîadu Blitter, przez co dane te muszâ znajdowaê sië w obszarze adresowalnym przez Blittera (czyli w pamiëci Chip). FBlit podmienia systemowe funkcje przetwarzajâce grafikë, które mogâ uûywaê Blittera, na analogiczne które uûywajâ procesora, dziëki czemu mogâ one operowaê na dowolnej pamiëci.

Po cóû miaîbym tego chcieê?

Przede wszystkim, 2MB pamiëci Chip to za maîo dla nowoczesnej Amigi (proszë sië nie ômiaê).

Pamiëê uûywana na ekrany musi byê pamiëciâ Chip, bo tylko do takiej ukîady generujâce obraz majâ dostëp, ale wszelkie inne dane graficzne sâ tylko dlatego przechowywane w Chip RAM'ie, aby systemowe funkcje uûywajâce Blittera mogîy je przetwarzaê. Wîaônie dla tych 'innych' (czyli nie-ekranów) danych graficznych, 2MB ograniczenie stanowi problem. Pojedynczy obrazek JPEG, po zdekodowaniu, moûe byê znacznie wiëkszy niû 2MB, przez co niemoûliwe staje sië jego wyôwietlenie, nawet w oknie na

otwartym juû ekranie. Albo jeûeli uûywasz NewIcons, moûe okazaê sië niemoûliwe wyôwietlenie zawartoôci duûego katalogu, poniewaû zabraknie Ci Chip RAM'u na wyôwietlenie wszystkich ikon. Takûe przeglâdarki HTML bardzo szybko zajmujâ caîâ dostëpnâ pamiëê Chip itd. itp.

Hej, jestem ELITA! Co to tak naprawdë jest?

Ah... Hmmm..... Cóû, FBlit skîada sië z kilku czëôci.

Nakîadki BltBitMap(), BitMapScale() i BltClear() pozwalajâ systemowi na uûywanie dyskretnych, niewyôwietlalnych bitmap znajdujâcych sië poza Chip RAM'em. Dziëki temu wiëkszoôê bitmap, elementów graficznych intuition, superbitmap itp. moûe byê przechowywana w Fast RAM'ie, jest to chyba najwaûniejsza czëôê FBlita.

Aby system mógî uûywaê rastportów/bitmap (i wiëkszoôci funkcji rysujâcych grafikë, np. Text(), RectFill(), Draw() itp.) poza obszarem pamiëci Chip, konieczne jest podmienienie kolejnych funkcji, jak BltTemplate(), BltPattern(), SetRast() i Draw(). Nie powstrzyma to caîkowicie systemu przed uûywaniem Blittera. Bëdzie on nadal uûywany przez np. funkcje Area...(), ale w takim wypadku jedynie na TmpRas, a nie na kaûdej bitmapie. Wprawdzie FBlit podmienia te funkcje (AreaEnd() i Flood()), dziëki czemu TmpRas zlokalizowany w Fast RAM'ie moûe byê obsîugiwany, ale trzeba mieê na uwadze ûe Blitter nadal bëdzie uûywany dla tych operacji.

Moûliwoôê obsîugi rastportów/bitmap bëdâcych poza pamiëciâ Chip niewiele daje, poniewaû bitmapa rastportu jest przewaûnie zwyczajnâ bitmapâ ekranu, na której rastport wystëpuje i dlatego musi ona znajdowaê sië w pamiëci Chip. Koniecznoôê ta zwiâzana jest z nastëpnâ czëôciâ FBlita. Gdy wszystkie bitmapy sâ alokowane z pîaszczyznami bëdâcymi w Fast RAM'ie, wtedy bitmapy pomocnicze zaalokowane na zachodzâce na siebie obszary okien takûe bëdâ w Fast RAM'ie, wiëc w pewnych okolicznoôciach fragmenty rastportów teû mogâ koïczyê sië poza pamiëciâ Chip.

Posiadanie systemu mogâcego obsîugiwaê niewyôwietlalne bitmapy znajdujâce sië w Fast RAM'ie jest ciekawâ perspektywâ, szczególnie dla programistów, którzy wiedzâ jak wyciâgnâê z tego korzyôê. Zdolnoôê ta nie wpîywa jednak na dotychczas powstaîe oprogramowanie. Jeûeli programista bëdzie chciaî skorzystaê z tej moûliwoôci, bëdzie musiaî alokowaê wîasne bitmapy z pîaszczyznami znajdujâcymi sië w Fast RAM'ie, co nie jest caîkowicie "przyjazne dla systemu", wiëc wiëkszoôê prawdopodobnie nie bëdzie z tego korzystaê. Aby zmusiê program do uûywania bitmap bëdâcych w Fast RAM'ie, konieczne jest podmienienie AllocBitMap(), co umoûliwi tworzenie w pamiëci Fast bitmap które zostaîy zaalokowane bez znacznika #BMF_DISPLAYABLE. Takie zachowanie moûe byê dopasowane dla kaûdego zadania oddzielnie, wiëc program który bëdzie miaî z tym problemy, moûe nadal mieê bitmapy z pîaszczyznami znajdujâcymi sië w pamiëci Chip.

Jest jeszcze kilka innych nakîadek, które powinny zaradziê kilku konkretnym zagadnieniom/problemom.

AddBob() i RemIBob() mogâ zostaê podmienione, dziëki czemu wszelkie obrazy BOB'ów bëdâce w Fast RAM'ie sâ w locie kopiowane z powrotem do pamiëci Chip. Umoûliwia to uûywanie NewIcons w trybie RTG i bezpieczne

w tym trybie ich przesuwanie.

 QBSBlit() moûe zostaê podmieniona, dziëki czemu odwoîania z DrawGList()
obsîugiwane sâ przez emulator, zamiast przez sprzëtowy Blitter. Odnosi
sië to do ikon z OS3.5.

 OpenScreenTagList() moûe zostaê podmieniona, dziëki czemu próby
otwierania ekranów z bitmapami zawierajâcymi pîaszczyzny spoza pamiëci
Chip mogâ byê prawidîowo obsîugiwane. Moûe sië to zdarzyê, gdy zadania
zmuszone do uûywania dla bitmap Fast RAM'u próbujâ otworzyê ekran
korzystajâcy z bitmapy, która nie zostaîa zaalokowana poprzez
#BMF_DISPLAYABLE.

1.5 Wymagania

Minimalne wymagania FBlita to:

 Procesor 68020
 v39/OS3.0
 Trochë Fast RAM'u

 Dodatkowo, FBlitGUI wymaga MUI.

 Ukîady AGA sâ zalecane, ale FBlit powinien teû dziaîaê na ukîadach
OCS/ECS.

 Zaleca sië teû aby nie uûywaê FBlita w wypadku posiadania
karty graficznej, chyba ûe wiesz co robisz! FBlit jest, na pewne sposoby,
podobny do oprogramowania RTG (CGX/P96), przez co próby uruchamiania na
raz FBlita i prawdziwego systemu RTG, bëdâ w najlepszym wypadku
problematyczne.

1.6 Instalacja

Instalacja

 Skopiuj 'FBlit' i 'FBlitGUI' do 'C:'.
 Skopiuj 'fblit.library' do 'Libs:'.

 Ewentualnie, przenieô katalog z FBlitem tam gdzie chcesz.

 Dopisz poniûszâ linië do 'S:startup-sequence' gdzieô za miejscem, gdzie
przypisany zostaje 'ENV:'. Linia tuû przed 'BindDrivers' jest przewaûnie
najodpowiedniejsza.

 C:FBlit

 (lub <ôcieûka do katalogu FBlita>/FBlit)

 Jeûeli aktualizujesz starszego FBlita i chcesz uûywaê najnowszych

standardowych ustawieï, to musisz przed zresetowaniem usunâê
'ENVARC:fblit.cfg'. (Uwaga: aktualnie jest to jedyny sposób na
przywrócenie ustawieï standardowych!)

Problemy zwiâzane z instalacjâ

Najwaûniejsze jest to, aby nie uruchamiaê FBlita poprzez 'run'
(ani nic podobnego). Jeûeli tak zrobisz, to nakîadki nie zostanâ
zainstalowane w miejscu w startup-sequence, z którego uruchomiony zostaî
FBlit i przez to kolejnoôê uruchamianych nakîadek nie bëdzie mogîa byê
zachowana.

FBlit powinien byê uruchamiany przed czymkolwiek co moûe podmieniaê
systemowe funkcje zwiâzane z grafikâ, czyli np. MCP, NewIcons, CGX itd.,
w przeciwnym wypadku efekty dziaîania takich programów zostanâ anulowane
przez nakîadki FBlita.

Programy wyôwietlajâce obrazek podczas bootowania teû mogâ podmieniaê
systemowe funkcje graficzne i moûliwe, ûe bëdziesz musiaî zainstalowaê
nakîadkë 'PatchControl' (albo podobnâ), jeûeli bëdziesz chciaî uruchamiaê
FBlita po takim programie.

Nie zaleca sië mieszania skîadników FBlita (GUI, bibliotek) z róûnych
wersji, bo sâ one przewaûnie niekompatybilne. Odnosi sië to teû do plików
z konfiguracjâ ('ENVARC:fblit.cfg'). Wprawdzie wszystkie wersje FBlita
sâ w stanie odczytaê i (w wiëkszym, lub mniejszym stopniu) zrozumieê
pliki .cfg pochodzâce z innych wersji, ale nie zawsze bëdâ one caîkowicie
kompatybilne. Jest to najbardziej niebezpieczne, gdy konfiguracja jest
nowsza, niû FBlit (np. konfiguracja w V3 z FBlitem w V2), ale w drugâ
stronë teû mogâ byê problemy, np. konfiguracje dla starszych wersji mogâ
mieê ustawione FDraw na Discard Fast Data (prawidîowo), podczas gdy
nowsze wersje powinny byê ustawione na Process.

Zmienianie nazw plików wchodzâcych w skîad FBlita nie jest dobrym
pomysîem i moûe przynieôê dziwne rezultaty.

Aktualnie, FBlit nie wyôwietla ûadnych komunikatów o bîëdach. Nie
poinformuje Cië, jeûeli nie uda mu sië zainstalowaê.

1.7 Obsîuga

To Co Jest Bezpieczne

Dla wiëkszoôci osób standardowe ustawienia FBlita powinny byê
wystarczajâce, wiëc nie namawiam Cië do ich zmiany, chyba ûe wiesz co
robisz i co z tego wyniknie. Moûesz jednak chcieê wprowadziê nastëpujâce
zmiany w pozostaîych czëôciach systemu.

NewIcons
Po zainstalowaniu i uruchomieniu FBlita, moûesz bezpiecznie przeîâczyê
NewIcons w tryb RTG (zob. NewIcons prefs/docs). Nie próbuj tego robiê

bez uruchomionego FBlita, bo w takim wypadku przesuwanie ikon bëdzie powodowaê faîszowanie zawartoôci pamiëci!

Kolorowe ikony z OS3.5

OS3.5 teû moûe obsîugiwaê ikony w trybie RTG i te teû bëdâ pracowaê z FBlitem, chociaû musisz w tym celu zrobiê coô wiëcej. Bëdziesz potrzebowaî nowego programu Stephana Rupprechta 'WBCtrl' (zob. Aminet).

W pliku S:startup-sequence, znajdú linië 'LoadWB'. Zmieï jâ na...

```
LoadWB SIMPLEGELS
```

...i linië przed tym, wpisz...

```
WBCtrl IMT=ICONFAST
```

Weú pod uwagë, ûe na takiej konfiguracji efekt przeúroczystoôci towarzyszâcy przesuwaniu ikon zniknie.

MCP

Jeûeli uûywasz MCP to powinieneô wyîâczyê nakîadkë QuickDraw i wîâczyê QuickLayers. Nie musisz tego robiê, ale aktualnie taka konfiguracja bëdzie wspóîpracowaê z FBlitem najlepiej.

WorkBench/Multiview/DataTypy

Po uruchomieniu FBlita, powinieneô zauwaûyê ûe podkîady WorkBencha/okien nie uûywajâ pamiëci Chip, wiëc moûesz szaleê z tym do woli (na tyle, na ile pozwala Fast RAM). To samo dotyczy Multiviewa, jeûeli uûywasz go raczej w oknie zamiast na ekranie. Wprawdzie wszystko co uûywa datatypów moûe czerpaê z tej wîaôciwoôci korzyôê, ale naleûy braê pod uwagë ûe nie wszystkie datatypy mogâ uûywaê Fast RAM'u. Zwykîa ilbm.datatype (<v43) zawsze bëdzie uûywaê pamiëci Chip, wiëc moûesz chcieê jâ zmieniê, lub wykorzystywaê na podkîady obrazki w innym formacie graficznym.

Ekrany/kolory

Moûesz teû teraz zwiëkszyê iloôê kolorów z jakâ otwarty jest ekran Twojego WB. Jeûeli uûywasz trybów PAL/NTSC (15KHz), ekran 256 kolorowy powinien byê caîkiem uûywalny. Jeûeli uûywasz trybów 30KHz (dblPAL itp.), to powinieneô sië ograniczyê do 64 kolorów.

Ekrany nadal bëdâ uûywaê tyle pamiëci Chip ile poprzednio, wiëc jeûeli otworzysz ich duûo, to iloôê wolnego Chip RAM'u tak jak wczeôniej spadnie bardzo szybko. Ale, skoro moûesz uûywaê bardziej kolorowego ekranu WB, to praktyczniej jest pozwoliê programom uûywaê takiego ekranu, zamiast kazaê im otwieraê wszystko na wîasnym.

FText

Nakîadka FText Ricka Pratta dostëpna jest na Aminecie. FText powoduje generowanie tekstów w Fast RAM'ie zamiast w pamiëci Chip, dziëki czemu wyôwietlanie go jest zauwaûalnie szybsze.

To Co Nie Jest Bezpieczne

FBlitGUI
pozwala Ci na zmienienie konfiguracji FBlita. Moûesz teû z jego pomocâ spowodowaê zapisy w losowe obszary pamiëci, zdestabilizowaê system, zniszczyê dane zapisane na twardym dysku.... Krótko mówiâc lepiej sië nim nie bawiê, jednak dla tych którzy i tak bëdâ sië bawiê, poniûej podane sâ pewne (moûliwe ûe jedyne) uûyteczne zmiany, których mogâ dokonaê.

Tryby Include/Exclude
Standardowo FBlit dziaîa w trybie 'Include'. Jeûeli chcesz, to moûesz do listy Include dodaê wiëcej zadaï. Lub zamiast tego przeîâczyê go w tryb 'Exclude', co w aktualnej wersji FBlita moûe byê lepszym rozwiâzaniem. (zob.
Listy Zadaï
)

Kolorowe ikony z OS3.5 (ponownie)
Istnieje alternatywna metoda obsîugi ikon w trybie RTG z OS3.5, która pozwala zachowaê efekt przeúroczystoôci podczas ich przesuwania, jednak odbywaê sië to bëdzie kosztem prëdkoôci. Uûywajâc FBlitGUI...

zainstaluj i zaktywizuj 'QBSBlitPatch'
odinstaluj 'AddBobPatch' i 'RemIBobPatch'

Moûesz teraz bezpiecznie usunâê parametr 'SIMPLEGELS' (o ile jest) z komendy 'LoadWB' w pliku S:startup-sequence.

Komenda 'WBCtrl IMT=ICONFAST' jest nadal potrzebna do przeîâczenia w tryb RTG ikon z OS3.5 . Moûesz teû zamiast tego spróbowaê wpisaê 'WBCtrl IMT=FAST', co przyspieszy nieco przesuwanie ikon, jednak nie jest pewne czy z aktualnâ wersjâ FBlita jest to bezpieczne.

Weú pod uwagë, ûe ustawienia takie bëdâ równieû dziaîaê z bëdâcymi w trybie RTG ikonami NewIcons, jednak w takim wypadku ustawienia standardowe wydajâ sië byê odpowiedniejsze. Mogâ teû wystâpiê pewne efekty uboczne, ponieważ teraz BOB'y bëdâ obsîugiwane na przerwaniach przez CPU. Z tego co wiem nie spowodowaîo to jeszcze wiëkszych problemów, ale niestety nie jest to najlepsze rozwiâzanie i jeôli wpadnë na lepsze to QBSBlitPatch moûe znów zniknâê i zostaê zastâpiony czymô w rodzaju Add/RemIBobPatch.

Jeûeli zdecydujesz sië powróciê do oryginalnych ustawieï, to musisz pamiëtaê o przywróceniu parametru 'SIMPLEGELS' i upewnij sië, ûe WBCtrl pracuje w trybie 'IMT=ICONFAST' (nie 'IMT=FAST'), w przeciwnym wypadku moûesz spodziewaê sië samych zîych rzeczy (konkretnie, ubytków Chip RAM'u, i zapisów w losowe obszary pamiëci Chip).

Ikony Ogólnie
Aby wszystko byîo jasne (ha!), istniejâ trzy moûliwoôci skonfigurowania FBlita pod kâtem przesuwania ikon/BOB'ów.

AddBobPatch/RemIBobPatch zainstalowane, QBSBlitPatch odinstalowany. Tak jest standardowo. Umoûliwia to przesuwanie w trybach RTG ikon NewIcons i ikon z OS3.5 (pod warunkiem, ûe uûywasz LoadWB z parametrem SIMPLEGELS, i ûe tylko icon.library (bez workbench.library) jest w trybie RTG). Bëdzie to teû dziaîaê jeûeli nie uûywasz ikon w trybie RTG.

QBSBlitPatch zainstalowany, AddBobPatch/RemIBobPatch odinstalowane. Ta konfiguracja równieû umoûliwia obsîugë w trybach RTG ikon NewIcons i ikon z OS3.5 (prawdopodobnie bez ûadnych warunków), jednak bëdzie sië to odbywaê wolniej. I znów, bëdzie to teû dziaîaê jeôli nie uûywasz ikon w trybie RTG.

AddBobPatch/RemIBobPatch i QBSBlitPatch odinstalowane. Na takiej konfiguracji nie moûna bezpiecznie przesuwaê ikon w trybie RTG, ale w celu posiadania jak najmniejszej iloôci nakîadek, moûesz jej uûywaê jeûeli nie masz OS3.5 lub NewIcons (albo jeôli po prostu nie chcesz uûywaê ikon w trybie RTG(?!)).

Wszelkie inne ustawienia tych trzech nakîadek sâ bezuûyteczne.

Prëdkoôê

Po przeczytaniu tego wszystkiego, wîaôciwie nic juû nie moûesz zrobiê aby cokolwiek przyspieszyê. Chociaû, w zaloûnoôci od sprzëtu jakiego uûywasz, spróbuj ustawiê FBltTemplate i FBltPattern na Process All Chip Data, moûe to przyspieszyê pewne operacje zwiâzane z grafikâ.

1.8 Róûne Problemy

Ogólnie

Wiëkszoôê problemów zwiâzanych z FBlitem wynika z niepoprawnych instalacji, lub z oddziaîywaï innych nakîadek (np. MCP, VisualPrefs). Moûe to wywoîaê wiele róûnych efektów, wybrakowanie grafiki, brak gadûetów w oknach, zniszczenie zawartoôci ekranu WorkBencha, okaleczenie ciaîa itd. itp.
(zob.
Problemy zwiâzane z instalacjâ
)

Linie

Bîëdy w rysowaniu linii sâ nastëpnym najczëôciej spotykanym problemem. Wynikajâ one z niedokoïczonej nakîadki podmieniajâcej funkcjë Draw() i chociaû wyglâda to brzydko, to jest to bezpieczne.
(zob.
FDraw
'Uwagi' aby dowiedzieê sië wiëcej)

Birdie/Stos

Nakîadki FBlita mogâ powodowaê zwiëkszanie wymagaï dotyczâcych wielkoôci stosu w niektórych funkcjach systemowych, co moûe powodowaê problemy. Kombinacja Birdie i FBlita moûe wystarczyê do uniemoûliwienia poprawnego dziaîania niektórych programów i moûliwe, ûe aby temu zapobiec bëdziesz musiaî uûywaê nakîadki 'StackAttack' (lub podobnej).

Wordworth7
WW7 moûe przestaê wyôwietlaê teksty jeûeli jego bitmapy bëdâ promowane do Fast RAM'u. Moûesz temu zaradziê ustawiajâc w tooltypie ikonki 'Wordworth' parametr 'PICASSO=TRUE'.

To i tamto wciâû uûywa pamiëci Chip
Nawet jeûeli uruchomisz FBlita w trybie Exclude, niektóre programy nadal bëdâ bez potrzeby uûywaê pamiëci Chip. Powodów moûe byê duûo i niewiele moûna z tym zrobiê. Programy mogâ tworzyê swoje wîasne bitmapy w pamiëci Chip, lub mogâ zawieraê grafikë wewnâtrz znajdujâcych sië w ich plikach wykonywalnych segmentów îadowanych do Chip RAM'u itd. itp. Jakkolwiek, jeûeli taki program uûywa datatypów, to moûliwe ûe któryô z nich (np. ilbm.datatype v39) moûe, nawet z FBlitem, uûywaê tylko pamiëci Chip.

Pamiëtaj teû, ûe Chip RAM nie tylko jest uûywany na grafikë, lecz teû na np. bufory audio.

OS3.5 i ubywanie pamiëci Chip
Jeûeli spróbujesz uûywaê ikon z OS3.5 w trybie RTG ze standardowo ustawionym FBlitem to zauwaûysz, ûe zacznie Ci ubywaê pamiëci Chip, chyba ûe dla LoadWB ustawisz parametr SIMPLEGELS.
(zob.
Obsîuga
)

Naleûy teû braê pod uwagë, ûe niezwiâzane z czymkolwiek ubywanie pamiëci Chip pod OS3.5 i tak wystëpuje, przynajmniej do pakietu uzupeîniajâcego BoingBag#1 (i wiâcznie z nim).

128 Kolorowe Ekrany
Ze 128 kolorowymi ekranami jest coô dziwnego, przynajmniej na niektórych Amigach 1200. Wyôwietlana grafika moûe byê przekîamana. Nie wiem z czego to wynika i nie potrzeba FBlita aby to spowodowaê. Stanie sië to nawet na czystej instalacji systemu, wiëc wyglâda na to, ûe jest to bîâd w systemie lub wada sprzëtowa.

1.9 Kto To Zrobiî?

Aktualnie caîy kod i opracowanie sâ dzieîem Stephena Brookesa.

Kontakt: sbrookes@tpec.u-net.com

Moje podziëkowania dla nastëpujâcych ludzi...

Artur Chlebek za polskâ dokumentacjë (z pomocâ Przemysîawa Gruchaîy i Mikoîaja Caîusiïskiego).

Evan Tuer za oryginalnâ ikonë MWB.

Phil Vedovatti i Luca Longone za ikony NewIcons.

I kolejnym osobom za róûne testy, wyszukiwanie bîëdów, wsparcie, zachëcanie itp.

Luca Longone
Rick Pratt
Ian Greenway
Przemysîaw Gruchaîa
Piotr Powlow
Marco De Vitis
Artur Chlebek
Jess Sosnoski
Matt Sealey
Evan Tuer
Gary Colville
James L Boyd
Colin Wenzel
Iain Barclay
Oliver Borrmann

I wszystkim innym którzy napisali do mnie w sprawie FBlita...

1.10 Historia

Zmiany od v2.63

3.66

FAllocBitMap
- tworzenie bitmap mogîo sië nie udawaê gdy podawane byîy niewîaôciwe parametry, faîszujâc przy tym losowe obszary pamiëci

FBlit
- powinien teraz odmawiaê uruchomienia jeûeli system jest <v39, jeûeli nie ma wolnego Fast RAM'u, lub jeôli procesor jest <68020, zamiast zawieszaê system (dziëki Mikoîaj Caîusiïski)

3.64

FAreaEnd
- nowa(stara) nakîadka obsîugujâca TmpRas bëdâcy w Fast RAM'ie

FFlood
- tak jak FAreaEnd

```
 FAllocBitMap
  - zostaîa rozbudowana i teraz alokuje bitmapy samemu. Gîównie
    umoûliwiîo to usuniëcie FAllocMem, ale równieû powinno byê to teraz
    bardziej kompatybilne z pamiëciâ wirtualnâ
  - zostaîa dodana opcja umoûliwiajâca wybór 'typu' pamiëci uûywanej na
    pîaszczyzny bitmap
  - zostaîa dodana ciekawa, naiwna/niebezpieczna/bezuûyteczna opcja
    umoûliwiajâca tworzenie wyôwietlalnych bitmap w pamiëci Fast

 FAllocMem
  - zniknëîa, poniewaû staîa sië zbëdna dziëki zmianom dokonanym w
    FAllocBitMap

 FBltBitMap
  - znów wróciîa bezpoôrednia obsîuga bitmap typu Interleaved (na liczne
    proôby (Albina))
  - zostaî znaleziony stary, gîupi bîâd, powodujâcy
    (rzadkie/niewystëpujâce) problemy podczas operacji odwrotnego
    kopiowania
  - procedura niewyrównanego kopiowania zostaîa przepisana, dziëki czemu
    staîa sië w ~100% efektywna pod wzglëdem iloôci odwoîaï do pamiëci
    (o ~33% mniej odwoîaï)
```

3.56

```
 FBltBitMap
  - jest w trakcie generalnej przeróbki...
  - zniknëîy 'nieîadne' procedury kopiowania. Wszystkie operacje sâ
    teraz wykonywane 'îadnie' (tzn. caîy czas jest w trybie 'Always
    Pretty')
  - funkcja kopiowania ma nowâ podprocedurë obsîugujâcâ w specyficzny
    sposób maîe operacje (<33bity)
  - zniknëîa opcja sprawdzania stosu
  - zniknëîa bezpoôrednia obsîuga bitmap typu Interleaved(!)

 FAreaEnd
  - zniknëîa, poniewaû staîa sië zbëdna (miejmy nadziejë), moûe jednak
    kiedyô powróciê w czëôciowo funkcjonujâcej formie...

 QBSBlitPatch
  - nowa nakîadka obsîugujâca BOB'y. Jest to odpowiednik AddBobPatch i
    RemIBobPatch, który moûe byê uûyteczny z ikonami z OS3.5 bëdâcymi w
    trybie RTG. Funkcjonuje ona identycznie jak FDrawGList i ûeby
    dziaîaîa musiaî powróciê stary emulator Blittera
```

3.51

```
 FBltBitMap
  - znaleziono i usuniëto bîâd. W trybie 'Always Pretty' pewne maîe
    operacje (<33bity) mogîy byê wyôwietlane o jedno dîugie sîowo na
    prawo od prawidîowej pozycji
  - dodano opcjë sprawdzania stosu
```

FAllocMem
 - zmieniîa sië tak, ûe moûe byê teraz bezpiecznie (miejmy nadziejë) instalowana po muguardianangel(MGA). Naleûy jednak braê pod uwagë, ûe jeûeli uruchomisz MGA po FBlicie, to promowanie bitmap zostanie wyîâczone! Moûesz wznowiê promocje poprzez odinstalowanie i ponowne zainstalowanie nakîadki FAllocMem (bëdzie to wymagaê programu 'PatchControl' lub podobnego)

FBltTemplate
 - uuups! Naprawiono paskudny bîâd. Gdy byîa ustawiona na Process Chip & Fast Data, FBltTemplate nie mogîa zablokowywaê layerów

FBltPattern
 - jest skoïczona, nareszcie. (Naprawiîo to teû maîy bîâd w wyrównywaniu wzorków, wystëpujâcy w pewnych okolicznoôciach w starej wersji)
 - jak z FBltTemplate, jeûeli ustawiona na Process Chip & Fast Data, nie bëdzie marnowaîa czasu na klasyfikowanie danych, takûe na takiej konfiguracji statystyki bëdâ nieprawidîowe

3.47

FBlit
 - zmieniîa sië metoda klasyfikacji RAM'u, dotyczy to teû fblit.lib

FAllocBitMap
 - czyôci #BMF_STANDARD (znowu). Uwaga: nie byîo to potrzebne poniewaû znacznik ten jest zbëdny w strukturze bitmap. Jest on tylko wartoôciâ zwrotnâ z GetBitMapAttr()

FBltPattern
 - ignoruje rp_AreaPtrn bëdâcy w pamiëci Fast (jeszcze raz)
 - usuniëto niebezpieczny bîâd (dziëki Luca)

FDrawGList
 - zniknëîa

AddBobPatch
 - nowa nakîadka przenoszâca obrazy BOB'ów z pamiëci Fast do Chip

RemIBobPatch
 - robi porzâdek po AddBobPatch

3.40

FBlit
 - fblit.library moûe byê teraz przechowywana w tym samym katalogu co FBlit

FBlitGUI
 - usuniëto bîâd zwiâzany z duûymi listami zadaï (dziëki Luca)

FAreaEnd

- poprawiono dymki pomocy ;)
- zmieniîy sië parametry Discard, teraz tylko odwoîania z TmpRas bëdâcych w Fast RAM'ie bëdâ odrzucane (których i tak nie powinno byê)

OSTLPatch
- nowa nakîadka (podchwytliwie nazwana) zwiâzana z problemami multiviewa/datatypów

3.36

FBlit
- usuniëto hity Enforcera generowane gdy FBlit nie mógî otworzyê bibliotek (dziëki Marco)

FBlitGUI
- usuniëto hity Enforcera generowane gdy uruchomiono FBlita z odinstalowanymi nakîadkami (znowu dziëki Marco)

FBltTemplate
- napisana od nowa (teraz uûywa fblit.library zamiast BlitEm)
- jeûeli ustawiona na Process Chip i Fast Data, nie bëdzie juû marnowaîa czasu na klasyfikowanie danych. Tak ûe na takiej konfiguracji statystyki bëdâ nieprawidîowe!

FBltPattern
- czëôciowo przepisana tak ûeby uûywaîa biblioteki... jeszcze nieskoïczona

FAreaEnd
- faîszowaîa wartoôê zwrotnâ funkcji (dziëki Luca)

FAllocMem
- napisana od nowa bez ûadnego konkretnego powodu. Statystyki zostaîy caîkowicie usuniëte!

FText
- zostaîa usuniëta, i (prawdopodobnie) nie wróci

3.32

- wszystko sië zmieniîo
- bîâd w nakîadce zostaî znów naprawiony. FBlit nadal by sië zawieszaî, jeûeli byîby uruchamiany z gîównego katalogu urzâdzenia, lub z urzâdzenia z odstëpem w nazwie ('Ram Disk:')

FDraw
- czëôciowo dziaîa. Nie obsîuguje jeszcze linii wzorkowanych i uzupeîniajâcych

1.11 FBlitGUI

Ogólnie

GUI FBlita moûe zostaê uruchomione z WorkBencha poprzez podwójne klikniëcie na ikonie FBlita, lub z CLI poprzez wywoîanie FBlita po raz drugi (np. 'c:FBlit'). Bezpoôrednie przywoîanie FBlitGUI teû moûe dziaîaê, ale nie zaleca sië tego. Naleûy braê pod uwagë ûe GUI moûna uûywaê tylko wtedy, gdy sam FBlit jest juû uruchomiony.

GUI FBlita jest w sumie caîkiem podobne do wielu innych edytorów preferencji bazujâcych na MUI, jednak jest w nim kilka istotnych róûnic!

Niemal wszystkie zmiany dokonywane w konfiguracji zaczynajâ dziaîaê od razu, nie ma wiëc gadûetu 'Test'.

Gadûet 'Quit' odnosi sië do samego FBlita, nie do jego GUI. Usuniëcie FBlita z pamiëci nie jest juû bezpieczne, wiëc nie powinieneô nigdy uûywaê tego gadûetu (a ja powinienem go usunâê).

Nie ma menu, i nie istnieje îatwy sposób przywrócenia ustawieï standardowych. Aktualnie moûna to osiâgnâê tylko poprzez usuniëcie 'ENVARC:fblit.cfg' i reset.

Inne gadûety funkcjonujâ w sposób, którego mógîbyô oczekiwaê...

'Save' zapisze konfiguracjë na staîe. Aktualna, i przyszîe uruchomione kopie FBlita bëdâ uûywaê tej konfiguracji.

'Use' nie zapisze konfiguracji. Tylko aktualnie uruchomiona kopia FBlita bëdzie z niej korzystaê.

'Cancel' przywróci konfiguracjë sprzed uruchomienia GUI.

Gadûet zamykajâcy okno dziaîa jak 'Cancel'.

Wiëkszoôê gadûetów wyôwietli dymki 'pomocy', jeûeli kursor bëdzie sië nad którymô znajdowaî wystarczajâco dîugo, i jeûeli nie wyîâczyîeô tej opcji. To czy faktycznie sâ one pomocne czy nie to juû inna sprawa.

Niebezpieczeïstwo!

Moûe juû wystarczajâco czësto o tym mówiîem, ale powtórzmy...

Uûywajâc GUI, bardzo îatwo skonfigurowaê FBlita w taki sposób ûe zacznâ wystëpowaê nielegalne zapisy w losowe obszary pamiëci Chip. Jest to bardzo niebezpieczne, na dodatek moûe sië to odbywaê bezobjawowo. Zapisy takie mogâ spowodowaê zawieszenie systemu, ale moûliwe jest teû przekîamanie danych które potem zostanâ zapisane na dysk, co moûe wiâzaê sië ze stratâ efektów ciëûkiej pracy, a w pewnych okolicznoôciach moûliwe jest nawet zniszczenie struktur systemu obsîugi plików na dysku!!

Te nielegalne zapisy wystâpiâ, gdy oryginalne systemowe funkcje które FBlit podmienia, majâ do czynienia ze strukturami danych bëdâcymi poza pamiëciâ Chip. Dlatego teû funkcje te FBlit podmienia w pierwszej

kolejnoôci.

Tak wiëc powinieneô bezwzglëdnie unikaê odinstalowywania i dezaktywowania nakîadek, chyba ûe w dokumentacji pisze inaczej. Poza tym, nie zmieniaj 'Fast Data Options' nakîadek, powinny one byê zawsze ustawione na 'Process' (lub 'Discard', jeûeli nie istnieje opcja 'Process').

1.12 Listy Zadaï

FBlit uûywa list z nazwami zadaï programów, które zostanâ ↩ zmuszone do
uûywania bitmap bëdâcych w pamiëci Fast. Jeûeli bëdziesz chciaî aby wiëcej programów uûywaîo dla bitmap Fast RAM'u, to moûesz to zrobiê na dwa sposoby.
(zob. GUI
FAllocBitMap
aby dowiedzieê sië wiëcej)

Tryb Include
Ten tryb jest ustawiony standardowo, i wymusi do uûywania dla bitmap Fast RAM'u tylko te zadania, które sâ na liôcie Include. W trybie tym lista Exclude jest ignorowana. Wadâ trybu Include jest to, ûe lista ta moûe staê sië bardzo duûa, a i tak moûesz nie wyîapaê wszystkich programów, które mogîyby byê bezpiecznie promowane.

Tryb Exclude
W tym trybie, wszystkie zadania zostanâ zmuszone do uûywania dla bitmap Fast RAM'u z wyjâtkiem tych, które sâ na liôcie Exclude. Lista Include jest ignorowana w tym trybie. Jest to bardziej niebezpieczne niû tryb Include, ponieważ program którego nie moûna bezpiecznie promowaê moûe zawiesiê Twój system zanim zorientujesz sië który to program!

Dodawanie Zadaï
Dodawanie zadaï w trybie Include, lub uûywanie trybu Exclude, jest generalnie niebezpieczne. Niektóre programy nie mogâ bezpiecznie uûywaê bitmap bëdâcych poza pamiëciâ Chip, i jeûeli bëdziesz te programy promowaî to mogâ one faîszowaê zawartoôê pamiëci! Przewaûnie, dotyczy to starszych programów, mniej lub bardziej przyjaznych dla systemu które bezpoôredio odwoîujâ sië do ukîadów Amigi w celu obsîugi bitmap, ale moûliwe jest teû wystâpienie problemów w nowszych programach, przyjaznych dla systemu (nawet przyjmujâc ûe sam FBlit jest pozbawiony bîëdów) np. mogâ one alokowaê pamiëê dla bitmap i uûywaê jej w innych celach (zakîadajâc ûe bëdzie to pamiëê Chip), lub umieszczaê w wolnych obszarach pamiëci Chip jakieô waûne dane.

Bez wzglëdu na tryb, proces dodawania zadaï jest taki sam.

Proste wpisanie nazwy programu do gadûetu tekstowego nie jest dobrym pomysîem z kilku powodów. Po pierwsze, 'nazwy zadaï' które FBlit

rozpoznaje sâ nazwami uûywanymi przez aktualne struktury zadania/procesu, a te niekoniecznie muszâ byê takie same (ani nawet podobne) do nazw samych programów. Po drugie, FBlit moûe wpîywaê tylko na zadania które wzywajâ AllocBitMap() ûâdajâc nie-wyôwietlalnej bitmapy, a wiele zadaï nie robi tego samemu.

W celu obejôcia tych problemów, FAllocBitMap ma opcjë 'Task Logging'. Jeûeli jest ona wîâczona, to nazwy zadaï które efektywnie mogâ zostaê dodane do listy zadaï, zostanâ przechowane w logu zadaï gdy wezwâ AllocBitMap(). Moûesz je potem wybraê z menu na stronie list zadaï w GUI. Zadania które nie pokazujâ sië na logu zadaï nie mogâ byê teû promowane do Fast RAM'u, i posiadanie takiego zadania na liôcie jest zwykîâ stratâ czasu. Naleûy jednak braê pod uwagë, ûe zadania które juû sâ na aktualnie aktywnej liôcie nie pojawiâ sië w logu!

Wiëc, poniûej znajduje sië proces dodawania zadaï do list... (bëdzie sië to wydawaê skomplikowane, ale w rzeczywistoôci tak nie jest, naprawdë, czy mógîbym Cië okîamywaê?)

Wîâcz 'Task Logging' funkcji FAllocBitMap (jeûeli juû wîâczyîeô, to pamiëtaj aby opuôciê GUI poprzez 'Use' albo 'Save', nie 'Cancel' ani gadûet zamykajâcy okno)

Upewnij sië ûe GUI FBlita jest zamkniëte. (aktualnie, z powodu lenistwa, log zadaï w GUI jest uzupeîniany tylko podczas uruchamiania!)

Teraz uruchom i uûywaj interesujâcego Cië programu. (jeûeli przez caîy czas byî uruchomiony to moûliwe ûe bëdziesz musiaî z niego wyjôê i uruchomiê go ponownie, aby zmusiê go do re-alokacji jego bitmap)

Ponownie uruchom GUI FBlita, i przejdú do odpowiedniej listy FAllocBitMap (czyli 'Include List' dla trybu Include, 'Exclude List' dla trybu Exclude).

Gadûetem bëdâcym po prawej stronie gadûetu tekstowego wîâcz menu i kliknij dwukrotnie na najlepiej pasujâcej nazwie zadania.

Moûliwe ûe bëdziesz musiaî trochë poeksperymentowaê jeûeli nie jest pewne która nazwa zadania odpowiada interesujâcemu Cië programowi.

Zmiany dokonane w listach zadaï nie bëdâ dawaîy rezultatu dopóki nie wyjdziesz z GUI poprzez 'Use lub 'Save, i nie wpîynâ na dodane/usuniëte zadania dopóki zadania te nie zostanâ zamkniëte i uruchomione ponownie.

Zadania na listë Exclude
Poniûej podane sâ zadania, które jeûeli sâ promowane, sprawiajâ problemy z FBlitem. Powinieneô dodaê je do listy zadaï Exclude jeûeli chcesz uûywaê trybu Exclude. I unikaê dodawania ich do listy Include, jeûeli uûywasz trybu Include.

'V'
- wyglâda na to ûe problemy zwiâzane z tym zadaniem zostaîy rozwiâzane poczâwszy od Voyagera w wersji 3.1, ale dla tych którzy nadal uûywajâ starszych wersji...

  Jest to gîówne zadanie Voyagera, i niestety aktualnie jest ono

niekompatybilne z FBlitem. Promowanie tego zadania bëdzie sprawiaê problemy gdy cache'owane obrazki sâ prze-skalowywane. Jakkolwiek, jest to dosyê rzadkie, i moûesz jednak chcieê go promowaê gdyû promocja moûe zredukowaê iloôê zawieszeï Voyagera zwiâzanych z innymi czynnikami. Moûe to teû dotyczyê, lub nie, starszych wersji Voyagera (<V^3).

'DPaint'
- dla niektórych wersji DPainta (przynajmniej dla wersji 5) zadanie to powinno znaleúê sië na liôcie Exclude. Wyglâda na to ûe uûywa on Blittera bezpoôrednio. Inne wersje DPainta mogâ pracowaê prawidîowo (lub nie).

Dodatkowo, dopóki FDraw jest nieskoïczony, moûesz z powodów estetycznych chcieê dodaê do listy Exclude programy które przejawiajâ problemy z rysowaniem linii (przewaûnie programy graficzne, zegarki analogowe).

1.13 Nakîadki

Ogólny opis wszystkich nakîadek znajduje sië pod koniec Co To Jest
.

Nakîadki FBlita moûna skonfigurowaê na wiele sposobów, jednak w wiëkszoôci inne ustawienia niû standardowe sâ bezuûyteczne dla normalnego uûytkownika. Niektóre opcje istniejâ gîównie dla celów testowych/developerów, i zabawa nimi jest potencjalnie niebezpieczna.

Instalacja Nakîadek

Chip/Fast Data Options

Informacje i Statystyki

FBltBitMap

FBltClear

FBltTemplate

FBltPattern

FBitMapScale

FFlood

FAllocBitMap

FSetRast

FAreaEnd

FDraw

OSTLPatch

AddBobPatch

RemIBobPatch

QBSBlitPatch

1.14 Instalacja Nakîadek

Patch Installation

Instalacja nakîadek wyglâda tak samo dla wszystkich nakîadek i sprowadza sië do dwóch gadûetów:

Installed decyduje czy nakîadka jest aktualnie zainstalowana w systemie, czy nie. Jeûeli nie uûywasz nakîadki 'PatchControl' (lub podobnej) to moûe okazaê sië niemoûliwe odinstalowanie danej nakîadki jeûeli zostaîa ona przepisana przez innâ.

Wiëkszoôêi nakîadek FBlita nie moûna bezpiecznie odinstalowywaê! Jeûeli opis konkretnej nakîadki nie mówi inaczej, to nigdy nie powinieneô próbowaê jej odinstalowywaê.
(zob. w
FBlitGUI
'Niebezpieczeïstwo!')

Activated decyduje czy nakîadka jest aktywna, czy nie. Umoûliwia to wyîâczenie nakîadki gdy odinstalowanie jej staîo sië niemoûliwe z powodu przepisania jej przez innâ nakîadkë. Zdezaktywowanie nakîadki jest analogiczne do odinstalowania jej, i dlatego komentarz dotyczâcy niebezpieczeïstwa odinstalowywania nakîadek tutaj teû sië odnosi. Proszë wiëc, nie rób tego, chyba ûe opis danej nakîadki mówi ûe to jest bezpieczne, w przeciwnym wypadku ryzykujesz przekîamywaniem zawartoôci pamiëci Chip!

1.15 Chip & Fast Data Options

Chip/Fast Data Options

Wszystkie nakîadki które podmieniajâ funkcje Blittera posiadajâ 'Chip Data Options' i 'Fast Data Options'. 'Chip Data' odnoszâ sië do danych które sâ w pamiëci Chip, a 'Fast Data' do danych bëdâcych gdziekolwiek indziej (niekoniecznie w Fast RAM'ie w dosîownym znaczeniu).

Chip Data Options wpîywajâ na operacje w których wszystkie majâce znaczenie dane sâ w pamiëci Chip, i które jako takie mogâ byê bezpiecznie przetwarzane przez oryginalne funkcje Blittera. W wypadku funkcji zwiâzanych z rastportami, powyûsze odnosi sië tylko do sytuacji w których wszystkie pîaszczyzny wszystkich powiâzanych bitmap sâ w Chip RAM'ie. Dla nakîadek posiadajâcych të opcjë dostëpne sâ nastëpujâce ustawienia (dla pewnych nakîadek istniejâ jeszcze dodatkowe ustawienia):

Pass On oznacza ûe do wykonania danej operacji zostanie uûyta oryginalna funkcja.

Process lub Process All , zastosowane zostanâ nakîadki uûywajâce CPU. Ma to zastosowanie gîównie w celach testowych, ale w pewnych wypadkach uûycie procedur CPU zamiast funkcji Blittera moûe okazaê sië szybsze.

Fast Data Options wpîywajâ na operacje w których jakiekolwiek dane sâ poza pamiëciâ Chip, przez co nie mogâ one byê przetwarzane przez oryginalne funkcje wykorzystujâce Blitter. Odnosi sië to do rastportów gdy dowolna pîaszczyzna z dowolnej bitmapy jest poza Chip RAM'em (obojëtne jest czy dana operacja dotyczy tych pîaszczyzn, czy nie). Dla wszystkich nakîadek posiadajâcych të opcjë moûliwe sâ nastëpujâce ustawienia:

Pass On przepuszcza danâ operacjë do oryginalnej funkcji. Jest to bezwzglëdnie bardzo zîy pomysî! Nie uûywaj tego ustawienia, bo na pewno bëdziesz miaî do czynienia z przekîamywaniem zawartoôci pamiëci Chip!

Process powoduje przeprowadzenie operacji przez procedury CPU. Jest to jedyne uûyteczne ustawienie dla tej opcji.

Discard oznacza ûe dane operacje poprostu nie zostanâ wykonane. Moûe to byê, lub nie, niebezpieczne, ale jest to zwyczajnie bezuûyteczne. Najprawdopodobniej efektem bëdzie wybrakowana/sfaîszowana grafika.

1.16 Informacje i Statystyki

Info & Stats

Znaczenie statystyk jest nieudokumentowane, i ich nazwy mogâ byê mylâce. To co zawierajâ nie jest zbytnio interesujâce, ale poniûej sâ podane wîaôciwe dla wszystkich nakîadek oznaczenia i gadûety dotyczâce statystyk:

Informacja o wersji jest (przewaûnie) prawidîowa dla wszystkich

nakîadek.

 Adresy Original/Current/Patch takûe powinny byê wiarygodne (chociaû naleûy braê pod uwagë ûe niektóre programy typu 'PatchControl' mogâ te wartoôci przekîamywaê). 'Original Addr' jest adresem oryginalnej funkcji która zostaîa podmieniona (wartoôê ta nie ma znaczenia dopóki nakîadka nie zostaîa zainstalowana). 'Current Addr' jest aktualnym adresem funkcji. 'Patch Code' jest adresem nakîadki. Niezbyt uûyteczne, chociaû moûesz dziëki temu stwierdziê czy na przykîad dana nakîadka nie zostaîa przepisana przez jakâô innâ ('Current Addr' bëdzie inny niû 'Patch Code'), lub czy oryginalna funkcja zostaîa juû podmieniona ('Original Addr' nie bëdzie w ROM'ie).

 Gadûety Update/Reset. Odnoszâ sië do statystyk, o ile dana nakîadka je prowadzi. Reset ustawi wszystkie liczniki statystyk z powrotem na zero. Update uzupeîni wartoôci w licznikach na aktualne. Statystyki nie sâ uzupeîniane w czasie rzeczywistym poniewaû niektóre nakîadki mogâ byê wzywane podczas odôwieûania GUI, co powodowaîoby sprzëûenie zwrotne. Jakkolwiek, weú pod uwagë ûe uûywanie gadûetu Update teû moûe wpîywaê na wartoôci statystyk.

1.17 FBltBitMap

Podmienia:
BltBitMap()

Cel:
 Pozwala BltBitMap() operowaê na bitmapach których pîaszczyzny sâ poza pamiëciâ Chip tzn. poza zakresem adresowalnym przez ukîad Blitter. Wpîywa to teû na inne funkcje np. ClipBlit(), BltBitMapRastPort(), BltMaskBitMapRastPort()... i w konsekwencji na bardziej zîoûone rzeczy jak synchroniczne operacje na superbitmapach, rysowanie elementów graficznych intuition, ikony....

Jak:
 FBltBitMap jest caîkowitym podmiennikiem dla BltBitMap() który do wykonywania operacji uûywa CPU zamiast Blittera, chociaû tam gdzie to jest moûliwe/stosowne wciâû moûe byê uûywany Blitter.

 Aktualnie, 'proste' funkcje (kopiowanie, kopiowanie z odwracaniem, wypeînianie itd.) uûywajâ wîasnych 32bitowych procedur. 'Zîoûone operacje' ('wycinacz wzorków'...) sâ wykonywane na prostym 16bitowym, trój-kanaîowym emulatorze funkcji Blittera.

Opcje:

Instalacja Nakîadki:
 Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FBltBitMap!

```
                  Chip Data Options:
                   (standardowo - Pass On Complex)

  Opcje dodatkowe:

   Pass On Complex  przepuôci operacje, które wymagajâ wykonania funkcji
logicznych na danych úródîowych i docelowych, do oryginalnej BltBitMap().
Aktualnie uûycie dla takich operacji procedur CPU nie ma ûadnej przewagi
nad Blitterem.


                  Fast Data Options:
                   (standardowo - Process)



Uwagi:

 FBltBitMap ma kilka 'efektów ubocznych'...

 Dla wiëkszoôci operacji jest szybsza niû oryginaî poniewaû (przynajmniej
na Amigach z ukîadami AGA) CPU ma 32bitowy dostëp, podczas gdy Blitter
jest ograniczony do 16bitów. Tak wiëc CPU moûe mieê o poîowë mniej
roboty. Oprócz tego CPU moûe unikaê niepotrzebnych odwoîaï do pamiëci.
Z drugiej strony Blitter moûe pracowaê równolegle z CPU, niezaleûnie
wykonywaê operacje logiczne i dzieliê z nim szynë Chip.

 Dla danych typu Non-Interleaved procedury CPU zredukujâ migotanie
kolorów poniewaû wszystkie (zawierajâce dane) pîaszczyzny sâ kopiowane
linia po linii na raz, podczas gdy Blitter przemieszcza na raz tylko
jednâ pîaszczyznë. Migotanie nadal moûe wystëpowaê gdyû pîaszczyzny nie
zawierajâce danych sâ obsîugiwane osobno od pîaszczyzn z danymi. Tak
wiëc, bez podwójnego buforowania, efekty przechodzenia rastru przez
jeszcze rysowanâ linië bëdâ nadal widoczne.
```

1.18 FBltClear

```
                Podmienia:
 BltClear()


Cel:
 Pozwala BltClear() dziaîaê poza pamiëciâ Chip. Jest to konieczne
poniewaû BltClear() moûe byê uûywana na pamiëci która jest czëôciâ
bitmapy.


Jak:
 FBltClear jest w peîni funkcjonujâcym, uûywajâcym tylko CPU
podmiennikiem BltClear().
```

Opcje:

Instalacja Nakîadki:
Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FBltClear!

Chip Data Options:
(standardowo - Pass On Asynch)

Opcje dodatkowe:

Pass On Asynch przepuôci tylko asynchroniczne operacje które mogâ byê wykonane przez Blitter podczas gdy CPU robi coô bardziej interesujâcego (teoretycznie).

Fast Data Options:
(standardowo - Process)

Uwagi:

FBltClear jest zazwyczaj szybsza niû oryginalna BltClear(), poniewaû 32bitowy CPU ma mniej pracy niû 16bitowy Blitter, przynajmniej na 32bitowym systemie. Dla wezwaï asynchronicznych prawdopodobnie nadal lepiej jest uûywaê Blittera, tam gdzie to jest moûliwe.

Istnieje pewien interesujâcy (pewnie nie dla Ciebie ;) efekt uboczny FBltClear, zgîoszony przez Ricka Pratta. Jeûeli jest ona ustawiona na Process All Chip Data, to mogâ wystâpiê pewne przekîamania (np. na oryginalnym analogowym zegarku WB). Nie jest to oczywiôcie nic dobrego, ale nie jest to wina kodu. Nie wiadomo na pewno czym to jest powodowane, dlaczego daje takie objawy, i dlaczego nie wpîywa na caîy system. Wystëpuje to tylko na (niektórych) systemach rozbudowanych o 68030, i najprawdopodobniej wynika to z wady sprzëtowej, zwiâzanej z cache'ami, lub pewnie z czegoô z taktowaniem szyny. W prawdzie cache dla danych i tak nie powinny byê aktywne w pamiëci Chip, ale wyîâczenie ich rozwiâzaîo ten problem... Wspaniale.

1.19 FBltTemplate

Podmienia:
BltTemplate()

Cel:
Pozwala BltTemplate() funkcjonowaê poza pamiëciâ Chip. Jest to potrzebne do obsîugi bitmap rastportów spoza Chip RAM'u.

Jak:
 FBltTemplate jest w peîni funkcjonujâcym, bazujâcym na CPU
odpowiednikiem BltTemplate().

Opcje:

Instalacja Nakîadki:
Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FBltTemplate ↩
!

Chip Data Options:
(standardowo - Pass On)

Fast Data Options:
(standardowo - Process)

Uwagi:

Niemalûe jedynâ rzeczâ do jakiej uûywana jest BltTemplate(), jest
generowanie tekstu. Poniewaû procedury CPU sâ 32bitowe, i z powodu natury
tekstu (przewaûnie formuje szeroki prostokât), FBltTemplate jest
najczëôciej szybsza niû BltTemplate(), wiëc moûesz chcieê ustawiê 'Chip
Data Options' na 'Process'. Zaleûy to od sprzëtu, i moûe byê zîym
pomysîem na ukîadach OCS/ECS, ale îatwo moûesz to sprawdziê wykonujâc
jakikolwiek test prëdkoôci wyôwietlania tekstu (SysSpeed itp.). Testy
sprawdzajâce prëdkoôê CON: przewaûnie jednak nie wykazujâ ûadnego
przyrostu.

1.20 FBltPattern

Podmienia:
BltPattern()

Cel:
Pozwala BltPattern() funkcjonowaê poza pamiëciâ Chip. Jest to potrzebne
do obsîugi bitmap rastportów spoza Chip RAM'u.

Jak:
FBltPattern jest w peîni funkcjonujâcym, bazujâcym na CPU podmiennikiem
BltPattern().

Opcje:

Instalacja Nakîadki:
Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FBltPattern!

Chip Data Options:
(standardowo - Pass On)

Fast Data Options:
(standardowo - Process)

Uwagi

BltPattern() jest szeroko uûywana przez system dla wielu (prawdopodobnie dla wiëkszoôêi) odwoîaï generujâcych rastport. Moûesz stwierdziê ûe FBltPattern jest szybsza od BltPattern() w niektórych operacjach (np. RectFill()), ale równie dobrze w innych moûe byê wolniejsza, wiëc nie powiem Ci czy ustawiê jej 'Chip Data Options' na 'Process', czy nie...

1.21 FBitMapScale

Podmienia:
BitMapScale()

Cel:
Pozwala BitMapScale() funkcjonowaê poza pamiëciâ Chip. Jest to potrzebne do obsîugi dyskretnych bitmap z pîaszczyznami w Fast RAM'ie.

Jak:
FBitMapScale jest bazujâcym caîkowicie na CPU podmiennikiem BitMapScale(). Moûe teû odwoîywaê sië do BltBitMap() wiëc wymaga FBltBitMap.

Opcje:

Instalacja Nakîadki:
Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FBitMapScale !

Chip Data Options:
(standardowo - Pass On)

Fast Data Options:
(standardowo - Process)

Uwagi:

FBitMapScale jest rzadko uûywana, i nie wîoûyîem w niâ ûadnego wysiîku. W wiëkszoôci przypadków jest prawdopodobnie wolniejsza od oryginaîu.

Warto teû odnotowaê ûe FBitMapScale nie daje takich samych efektów swojego dziaîania jak oryginalna funkcja BitMapScale().

1.22 FFlood

Podmienia:
Flood()

Cel:
Pozwala Flood() operowaê na TmpRas nie bëdâcym w Chip RAM'ie.

Jak:
Nakîadka ta sprawdza dostarczony TmpRas. Jeûeli nie jest on w pamiëci Chip, to zostanie zaalokowany nowy TmpRas w Chip RAM'ie, i wtedy wezwana zostaje Flood(). Gdy Flood() zakoïczy operacjë, TmpRas z pamiëci Chip zostaje znów zwolniony.

Opcje:

Instalacja Nakîadki:
FFlood nie powinnna byê odinstalowywana lub dezaktywowana.

Uwagi:

Nakîadka ta nadal potrzebuje wystarczajâcej iloôci pamiëci Chip do wykonania oryginalnej funkcji, a operacje z TmpRas bëdâcym w Fast RAM'ie w rzeczywistoôci bëdâ wolniejsze od tych z TmpRas bëdâcym pierwotnie w pamiëci Chip.

Jedynym aktualnie znanym programem uûywajâcym TmpRas w Fast RAM'ie jest PPaint, gdy jest w peînym trybie RTG.

1.23 FAllocBitMap

Podmienia:
AllocBitMap()

Cel:
Nakîadka ta wymusza alokowanie danych graficznych w pamiëci Fast.

Jak:
W zaleûnoôci od konfiguracji, FAllocBitMap decyduje czy wezwane, przez dane zadanie, AllocBitMap() powinno byê promowane, czy nie. Jeûeli bitmapa ma zostaê zaalokowana w Fast RAM'ie, FAllocBitMap stworzy jâ samemu, w przeciwnym wypadku zostanie wezwana AllocBitMap().

Opcje:

Instalacja Nakîadki:
FAllocBitMap moûe zostaê caîkiem bezpiecznie odinstalowana, lub zdezaktywowana w dowolnej chwili. Zrobienie tego powstrzyma FBlita od promowania grafiki do Fast RAM'u.

Task Logging:
Jeûeli opcja ta jest wîâczona, nazwy zadaï które wzywajâ AllocBitMap() zostanâ przechowane w logu zadaï. Na zadania które nie chcâ sië ukazaê w logu, FAllocBitMap nie ma wpîywu i dlatego nie powinny one byê na listach zadaï.

Trzeba pamiëtaê ûe kopia logu zadaï uûywana przez FBlitGUI nie jest utrzymywana w czasie rzeczywistym, jest ona uzupeîniana/prawidîowa tylko w momencie przywoîywania GUI. Dlatego aby uzyskaê aktualnâ kopië logu bëdziesz musiaî GUI zamknâê i uruchomiê ponownie.

Jest jeszcze kilka rzeczy w logu zadaï o których naleûy wiedzieê. Zadania które juû sâ na aktualnie aktywnej liôcie nie ukaûâ sië w logu. Poza tym, log w przeciwieïstwie do list zadaï, rozpoznaje wielkoôê znaków, co oznacza ûe dane zadanie moûe ukazaê sië kilka razy z róûnymi wielkoôciami liter (np. 'Multiview' i 'multiview').

Task List Options:
FAllocBitMap ma dwa tryby zmuszania zadaï do alokowania bitmap w Fast RAM'ie.

Tryb Include jest (aktualnie) ustawiony standardowo, i jest bezpieczniejszy. W trybie tym do pamiëci Fast bëdâ promowane tylko te zadania które sâ na 'Include List'.

Tryb Exclude jest trochë bardziej ryzykowny. W tym trybie wszystkie zadania zostanâ zmuszone do uûywania Fast RAM'u, z wyjâtkiem tych z 'Exclude List'.

Bierz pod uwagë ûe w danej chwili tylko jedna z dwóch list zadaï jest aktywna. Dla trybu 'Include', uûywana jest tylko 'Include List', a 'Exclude List' jest ignorowana. I na odwrót w trybie 'Exclude'.

Anonymous Tasks:
Opcja ta definiuje sposób postëpowania z zadaniami bez nazwy, które jako takie nie mogâ byê obsîugiwane przez listy zadaï.

Pass On , zadania anonimowe bëdâ uûywaê Chip RAM'u (standardowo).

Promote , zadania te bëdâ uûywaê Fast RAM'u.

Displayable Bitmaps
Decyduje czy wyôwietlalne bitmapy powinny byê promowane do pamiëci Fast, czy nie.

Pass On , wyôwietlalne bitmapy bëdâ uûywaîy pamiëci Chip (standardowo).

Promote , wyôwietlalne bitmapy bëdâ uûywaîy pamiëci Fast. To nie jest dobry pomysî! Proszë, nie uûywaj tego ustawienia! Opcja ta istnieje tylko dla osób które chciaîyby napisaê sterownik wyôwietlajâcy ekrany bëdâce w Fast RAM'ie. Po ustawieniu tej opcji bez takiego sterownika, kaûdy nowo otwarty ekran bëdzie przedstawiaî ômieci.

Promotion Memory
Definiuje 'typ' pamiëci uûywanej na pîaszczyzny bitmap.

MEM_FAST , jest ustawiony standardowo, i oznacza ûe pîaszczyzny zawsze bëdâ uûywaê tylko pamiëci Fast! Nawet jeûeli nie bëdziesz juû miaî pamiëci Fast, i bëdzie jeszcze mnóstwo wolnego Chip (albo dowolnego nie-Fast) RAM'u, to i tak nigdy nie zostanie on uûyty na pîaszczyzny bitmap.

MEM_ANY , oznacza ûe na pîaszczyzny moûe zostaê uûyta dowolna pamiëê. Najpierw bëdzie uûywana 'najlepsza' dostëpna pamiëê, a inne typy 'publicznej' (tzn. nie wirtualnej) pamiëci zostanâ uûyte w razie potrzeby. Opcja ta powinna pracowaê prawidîowo (jeôli nie, to z powodu bîëdów w FBlicie), i byê ustawiona standardowo, nie zostaîa jednak dokîadnie przetestowana, wiëc (jeszcze) nie jest.

Lists
Strona 'Lists' pozwala na edytowanie list zadaï 'Include' i 'Exclude'. Zadania mogâ byê dodawane poprzez wpisywanie ich nazw do gadûetu tekstowego, lub przez wybieranie ich z menu task logu. Aby usunâê dane zadanie, zaznacz je na liôcie i wciônij gadûet 'Remove'.

Edytowanie list zadaï jest jedynâ czynnoôciâ w GUI która nie wpîywa na system w czasie rzeczywistym. Zmiany dokonane w listach zadaï zacznâ dziaîaê dopiero gdy wyjdziesz z GUI (poprzez 'Use' lub 'Save').

W przeciwieïstwie do logu zadaï, listy zadaï nie rozpoznajâ wielkoôci znaków. Wiëc pojedyïczy wpis ('multiview') bëdzie sië odnosiî do odpowiedniego zadania bez wzglëdu na wielkoôê liter w jego nazwie

('Multiview', 'MultiView').

Uwagi:

Nie gwarantujë ûe bitmapy stworzone przez FAllocBitMap bëdâ identyczne do tych stworzonych przez AllocBitMap().

Aktualnie, FAllocBitMap tworzy bitmapy o szerokoôci wyrównanej do 16bitów (lepiej by byîo do 32, ale powoduje to problemy z pewnymi bardzo popularnymi programami), wyôwietlalne bitmapy majâ szerokoôê wyrównanâ do 64bitów.

Minplanes sâ zaimplementowane, na tyle na ile sâ tego warte. Bitmapy typu Friend sâ obsîugiwane tak, jak to ma zastosowanie, ûe bitmapa zostanie przerobiona na typ Interleaved aby pasowaîa do tej typu 'Friend'. Bitmapy typu Interleaved majâ wielkoôê ograniczonâ do <1024 bajtów na linië, i <1024 na linie. bm_Pad jest ustawiony na zero, lub magicznâ wartoôê Interleaved. bm_Flags sâ ustawione na zero.

FreeBitMap() nie jest podmieniona. FAllocBitMap polega na jej funkcji w taki sposób, jak robi to graphics.library v39/40 (i nie musi w starszych, lub (tu raczej maîo prawdopodobne) przyszîych wersjach).

Naleûy odnotowaê ûe chociaû AllocBitMap() akceptuje parametry ULONG, to wyglâda na to ûe ignoruje ona wyûsze WORD co umoûliwiîo funkcjonowanie pewnym programom które przepuszczaîy te wartoôci przekîamane. Aby obsîugiwaê takie uszkodzone programy, FAllocBitMap maskuje wartoôci wejôciowe WORD dla 'sizex'/'sizey', i BYTE dla 'depth'.

1.24 FSetRast

Podmienia:
SetRast()

Cel:
Pozwala SetRast() pracowaê poza pamiëciâ Chip.

Jak:
Nakîadka ta do wykonania czynnoôci wzywa albo BltClear() albo BltBitMapRastPort(), wiëc obie te funkcje muszâ móc operowaê poza pamiëciâ Chip (tzn. FBltClear i FBltBitMap sâ wymagane).

Opcje:

Instalacja Nakîadki:

Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FSetRast!

Chip Data Options:
(standardowo - Pass On)

Fast Data Options:
(standardowo - Process)

1.25 FAreaEnd

Podmienia:
AreaEnd()

Cel:
Pozwala AreaEnd() obsîugiwaê TmpRas bëdâcy w Fast RAM'ie.

Jak:
Nakîadka ta sprawdza dostarczony TmpRas. Jeûeli nie jest on w pamiëci Chip, to zostaje zaalokowany nowy TmpRas w Chip RAM'ie, i wtedy wezwana zostaje AreaEnd(). Gdy AreaEnd() zakoïczy operacjë, TmpRas z pamiëci Chip zostaje znów zwolniony.

Opcje:

Instalacja Nakîadki:
FAreaEnd nie powinnna byê odinstalowywana lub dezaktywowana.

Uwagi:

Nakîadka ta nadal potrzebuje wystarczajâcej iloôci pamiëci Chip do wykonania oryginalnej funkcji, a operacje z TmpRas bëdâcym w Fast RAM'ie w rzeczywistoôci bëdâ wolniejsze od tych z TmpRas bëdâcym pierwotnie w pamiëci Chip.

Jedynym aktualnie znanym programem uûywajâcym TmpRas w Fast RAM'ie jest PPaint, gdy jest w peînym trybie RTG.

1.26 FDraw

Podmienia:
Draw()

Cel:
Pozwala Draw() pracowaê poza pamiëciâ Chip.

Jak:
FDraw jest w peîni bazujâcym na CPU odpowiednikiem Draw(). ...Moûe kiedyô bëdzie, jeôli kiedykolwiek jâ dokoïczë.

Opcje:

Instalacja Nakîadki:
Nigdy nie powinieneô odinstalowywaê lub dezaktywowaê FDraw!

Chip Data Options:
(standardowo - Process Hor)

Opcje dodatkowe:

Process Hor spowoduje rysowanie poziomych, nie-wzorkowanych linii, przez procedury CPU. Procedury CPU sâ znacznie szybsze od Draw() dla poziomych linii, ale mogâ byê wolniejsze dla wszystkich innych.

Fast Data Options:
(standardowo - Process)

Uwagi:

FDraw jest nieskoïczona co moûe powodowaê problemy! Gîównie bëdzie sië to objawiaê nie znikaniem niektórych linii. Jest to spowodowane brakiem obsîugi rysowania w trybie COMPLEMENT, jest to bezpieczne, chociaû niezbyt îadnie wyglâda. Efekty te na standardowych ustawieniach zdarzajâ sië rzadko, ale jeûeli ustawisz FDraw na Process All Chip Data, to zacznâ wystëpowaê znacznie czëôciej. Bëdâ teû czëstsze jeûeli FAllocBitMap jest ustawiona na tryb 'Exclude', poniewaû w takim wypadku wiëkszoôê operacji bëdzie wykonywana przez procedury CPU.

Aktualnie FDraw nie rysuje linii ukoônych identycznych jak Draw(). Efekty tego moûna zauwaûyê gdy linie sâ rysowane jedna obok drugiej, lub na granicah prostokâtów rysowanych przez Blitter (wskazówki zegarów).

1.27 OSTLPatch

```
                Podmienia:
 OpenScreenTagList()


Cel:
 Ten hack usiîuje powstrzymaê programy przed otwieraniem ekranów z
nie-wyôwietlalnymi bitmapami które mogîy zostaê promowane do Fast RAM'u.


Jak
 Jeûeli bitmapa jest wzywana poprzez OpenScreen/TagList(), nakîadka
sprawdzi czy jej pîaszczyzny sâ w pamiëci Chip. Jeûeli nie, spróbuje ona
zaalokowaê identycznâ bitmapë z ustawionym znacznikiem #BMF_DISPLAYABLE.
Jeûeli sië to nie uda, nakîadka zwraca bîâd. Jeôli sië uda, oryginalny
obraz jest kopiowany do nowej bitmapy poprzez BltBitMap(), i (co jest
nieco nieprzyjemne) zawartoôê struktur bitmap jest zamieniana. Nowo
zaalokowana bitmapa (z oryginalnâ definicjâ bitmapy) jest zwalniana.
Oryginalna bitmapa (z nowâ definicjâ bitmapy) zostaje uûyta dla wezwania
OpenScreenTagList().


Opcje:


                 Instalacja Nakîadki:
                 Tak dîugo jak reszta FBlita wykonuje swojâ pracë, odinstalowanie
OSTLPatch moûe byê bezpieczne. Rezultatem tego moûe byê otwieranie
ekranów w Fast RAM'ie, co jest bardzo rzadkie (aktualnie najczëstszym
napastnikiem jest Multiview). Ekrany takie najczëôciej bëdâ zawieraîy
same ômieci poniewaû ukîady generujâce obraz bëdâ wyôwietlaê jakieô
losowe obszary pamiëci Chip.



Uwagi:

 Jak juû mówiîem, dopóki reszta FBlita pracuje prawidîowo, otwieranie
ekranów w Fast RAM'ie jest bezpieczne. Po prostu ich nie zobaczysz, wiëc
nie jest to zbytnio pomocne. Jakkolwiek, jest to ciekawa moûliwoôê.
Teoretycznie, trzymanie ekranów w Fast RAM'ie jest moûliwe i do
wyôwietlania tego bëdâcego na wierzchu moûna by uûywaê prostego
'sterownika video'. Moûe z podwójnym buforowaniem? Odôwieûaniem MMU? Moûe
nie... Oczywiôcie byîyby pewne problemy... Jak zawsze, bardziej
odpowiednie byîoby napisanie odpowiedniego sterownika dla P96. Lub
oczekiwanie na to ûe system zacznie obsîugiwaê RTG.
```

1.28 AddBobPatch

```
                Podmienia:
```

```
 AddBob()


Cel:
 Wraz z
                 RemIBobPatch
                 hack ten jest konieczny w celu umoûliwienia
przesuwania ikon, gdy NewIcons sâ w trybie RTG.


Jak:
 Jeûeli obraz BOB'a jest poza Chip RAM'em, zostanie on z powrotem
skopiowany do automatycznie-powiëkszanego bufora w pamiëci Chip. Aby to
odzwierciedliê zmieniony zostanie wskaúnik obrazu BOB'a, i wezwana
zostanie AddBob().


Opcje:


                 Instalacja Nakîadki:
                 Nakîadka ta moûe zostaê bezpiecznie odinstalowana jeûeli nie  ↩
                    uûywasz
NewIcons (lub icon.library z OS3.5) w trybie RTG, lub jeûeli uûywasz

                 QBSBlitPatch
                . Jeûeli odinstalujesz AddBobPatch, musisz teû odinstalowaê
RemIBobPatch!



Uwagi:

 Hack Add/RemIBobPatch zostaî opracowany gîównie z myôlâ o RTG NewIcons.
Oficjalnie FBlit nie obsîuguje GELS bëdâcych poza pamiëciâ Chip.
```

1.29 RemIBobPatch

```
               Podmienia:
 RemIBob()


Cel:
 Wraz z
                 AddBobPatch
                 hack ten jest konieczny w celu umoûliwienia
przesuwania ikon, gdy NewIcons sâ w trybie RTG.


Jak:
 Zostaje wezwana RemIBob(). Wtedy, jeûeli obraz BOB'a jest w buforze w
pamiëci Chip, wskaúnik orazu BOB'a zostanie przywrócony na obraz BOB'a z
```

Fast RAM'u. Jeûeli byî to ostatni obraz w buforze, bufor ten zostanie
zwolniony.

Opcje:

Instalacja Nakîadki:
Nakîadka ta moûe zostaê bezpiecznie odinstalowana jeûeli nie ↩
uûywasz
NewIcons (lub icon.library z OS3.5) w trybie RTG, lub jeûeli uûywasz

QBSBlitPatch
. Jeûeli odinstalujesz RemIBobPatch, musisz teû odinstalowaê
AddBobPatch!

Uwagi:

Hack Add/RemIBobPatch zostaî opracowany gîównie z myôlâ o RTG NewIcons.
Oficjalnie FBlit nie obsîuguje GELS bëdâcych poza pamiëciâ Chip.

1.30 QBSBlitPatch

Podmienia:
QBSBlit()

Cel:
Pozwala na przesuwanie w trybie RTG ikon z OS3.5.

Jak:
Wezwania QBSBlit() z DrawGList() sâ przechwytywane, i bltnode/funkcja
zostaje zmodyfikowana w celu obsîuûenia jej na emulatorze Blittera.
Wtedy wezwana zostaje oryginalna QBSBlit().

Opcje:

Instalacja Nakîadki:
Aktualnie, standardowo nakîadka ta jest odinstalowana. Jeûeli jâ
zainstalujesz, musisz odinstalowaê AddBobPatch i RemIBobPatch (i na
odwrót).

Uwagi:

Dodatkowo nakîadka ta prawie w peîni obsîuguje GELS (z wyjâtkiem

'SimpleSprites') bëdâce poza pamiëciâ Chip, chociaû nie robi tego bezpoôrednio FBlit.

Emulator Blittera uûywany przez QBSBlitPatch nie jest zbyt szybki. Jest znacznie wolniejszy od sprzëtowego Blittera nawet na 060/50 i gdy wszystkie dane sâ w Fast RAM'ie.

Jest z tym zwiâzany pewien problem. Emulator Blittera dziaîa na przerwaniach, i przez to bëdzie je blokowaî na znacznie dîuûej niû normalna bltnode/funkcja. Nie zgîoszono jeszcze ûadnych zwiâzanych z tym efektów ubocznych, ale nie oznacza to ûe ich nie ma.

1.31 Programming

under contruction