Plananarama 2.2

The Planar Graphics Engine for Hollywood

Andreas Falkenhahn

Inhaltsverzeichnis

1 Allgemeine Informationen

1.1 Einführung

Das Plananarama-Plugin ermöglicht es Hollywood, auf planaren (palettenbasierten) Bildschirmen zu laufen. Damit ist es endlich möglich, Hollywood-Skripte auf einfachen AGA- oder ECS-Systemen ohne installierte Grafikkarte auszuführen - zum ersten Mal seit Hollywood 1.93 (veröffentlicht im März 2005)! Ab Version 2.0 benötigte Hollywood CyberGraphX oder Picasso96, um ausgeführt zu werden. Seitdem haben viele Leute um eine Wiederbelebung von Hollywoods Planar-Modul gebeten, und hier ist sie nun, eine echte Explosion aus der Vergangenheit! Dank der stark erweiterten Plugin-API von Hollywood 6 konnte dieses Feature vollständig im Plugin-Bereich implementiert werden. Sobald Plananarama installiert ist, werden alle Hollywood-Skripte wieder "automatisch" auf Palettenbildschirmen ausgeführt! Alle Auflösungen werden unterstützt - von 8-Bit LowRes bis zu 1-Bit Productivity SuperHighRes Interlace.

Ab Plananarama 2.0 ist es auch möglich, im Vollbildmodus direkten Zugriff auf die Palettenstifte des Bildschirms zu erhalten. Auf diese Weise können Sie direkt mit der Palette des Bildschirms zeichnen, ohne dass eine Neuzuordnung der Farben erforderlich ist, sodass das Zeichnen viel schneller wird. Außerdem ist es möglich, durch Wechseln der Stifte schöne Paletteneffekte zu erzeugen, z.B. ist es möglich, Farben fast ohne CPU-Last zu verblassen oder zu wechseln, da Plananarama im Paletten-Modus das Farbregister der Hardware direkt ändern kann, anstatt alles neu zeichnen zu müssen, wenn sich Farben ändern (wie es für echte TrueColour-Grafiken erforderlich ist).

Außerdem führt Plananarama 2.0 die Unterstützung für echte Hardware-Sprites ein. Bei der Verwendung von Hardware-Sprites wird die benutzerdefinierte Chip-Hardware des Amiga verwendet, um die Sprites zu zeichnen, wodurch Ihr Skript Sprites in kürzester Zeit fast ohne CPU-Last zeichnen kann.

All dies macht Plananarama zum ultimativen Plugin für klassische Amiga-Systeme ohne installierte Grafikkarte.

1.2 Lizenzvereinbarung

1.3 Anforderungen

Dieses Plugin benötigt mindestens Hollywood 6.0, da es die mit Hollywood 6.0 eingeführten Display- und Bitmap-Adapter-APIs verwendet. Einige Funktionen erfordern jedoch neuere Hollywood-Versionen. Hier ist eine Übersicht über die Funktionen, die nur mit neueren Hollywood-Versionen verfügbar sind:

- Unicode-Unterstützung: Erfordert Hollywood 7
- Unterstützung für `VanillaKey` und `OnDropFile`-Ereignis-Handler: Erfordert Hollywood 7
- Unterstützung für `OnRawKey-Ereignis-Handler`: Erfordert Hollywood 7.1
- Unterstützung für Menüs: Erfordert Hollywood 9
- Unterstützung für den speziellen Paletten-Modus von Plananarama: Erfordert Hollywood 9
- Unterstützung für Hardware-Sprites: Erfordert Hollywood 9

Die folgenden optionalen Komponenten können ebenfalls erforderlich sein:

- guigfx.library and render.library: Diese beiden Bibliotheken werden benötigt, wenn der Tag `PaletteMode` im `@REQUIRE`-Aufruf auf `False` gesetzt wurde (was auch der Standardwert ist). In diesem Fall wird Plananarama alle Grafiken auf die Palette des aktuellen Bildschirms neu zuordnen. Dies geschieht mit Hilfe von guigfx.library und render.library. Wenn Sie jedoch `PaletteMode` auf `True` setzen, benötigt Plananarama guigfx.library und render.library nicht, da die Grafiken nicht auf die Bildschirm-Palette abgebildet werden, sondern alle Grafiken mit den Stiften des Bildschirms gezeichnet werden.

Beachten Sie, dass render.library in zwei Varianten verfügbar ist: Es gibt einen v40- und einen v30-Zweig. Es wird empfohlen, die neueste Version des v30-Zweigs zu verwenden, da alle v40-Versionen C-Portierungen der Versionen des v30-Zweigs sind, welche alle in 68k-Assembler geschrieben sind. Die Assembler-Version (= v30-Zweig) von render.library ist viel schneller als die C-Version, weshalb Sie vielleicht lieber stattdessen die Assembler-Version verwenden möchten. Die neueste Assembler-Version von render.library ist v32.0. Wenn Ihnen also Leistung wichtig ist, sollten Sie diese Version verwenden. Außerdem benötigt der v40-Zweig von render.library eine FPU, was ein weiterer Grund dafür ist, stattdessen die neueste Version aus dem v32-Zweig zu verwenden.

- FBlit: Wenn der Tag `PaletteMode` im `@REQUIRE`-Aufruf auf `False` gesetzt wurde (was auch die Voreinstellung ist), wird dringend empfohlen, FBlit zu verwenden, weil sonst der Chipspeicher in kürzester Zeit aufgebraucht sein wird und Hollywood der Speicher ausgeht. Stellen Sie also sicher, dass Sie zuerst FBlit installieren und Hollywood zu seiner "Include"-Liste auf der Registerkarte "FAllocBitMap" hinzufügen. Wenn Hollywood in dieser Liste enthalten ist, wird FBlit in der Lage sein, Grafiken im Fast-RAM zu platzieren, was unbedingt benötigt wird, da 2 MB Chip-RAM für Hollywood sicherlich nicht ausreichen werden. Die Verwendung von FBlit hat auch den Vorteil, dass Hollywood die CPU zum Blittern verwenden kann, was ohnehin viel schneller ist als das Blittern auf höheren 68k-CPUs oder WinUAE. Wenn `PaletteMode` auf `True` gesetzt ist, ist FBlit nicht erforderlich, da Hollywood seine Grafikdaten im schnellen Speicher speichern kann. In diesem Fall sollten Sie jedoch BlazeWCP verwenden (siehe unten).
- BlazeWCP: Wenn der Tag `PaletteMode` im `@REQUIRE`-Aufruf auf `True` gesetzt wurde, speichert Hollywood alle Grafiken im Fast-RAM und zeichnet sie mit dem Befehl `WriteChunkyPixels()` aus der graphics.library. `WriteChunkyPixels()` ist jedoch unter OS3.1-3.9 sehr langsam, daher sollten Sie vielleicht BlazeWCP aus dem Aminet verwenden, um die Dinge zu beschleunigen. Gerüchte besagen, dass die Implementierung von `WriteChunkyPixels()` in den neuen klassischen AmigaOS-Versionen von Hyperion (3.1.4 und höher) einige Optimierungen erfahren hat, sodass BlazeWCP auf 3.1.4 und höher möglicherweise nicht mehr erforderlich ist, aber ich habe keine Benchmarks durchgeführt, daher kann ich nicht sagen, ob BlazeWCP mit den klassischen AmigaOS-Veröffentlichungen von Hyperion noch notwendig ist. Es wird definitiv mit dem klassischen AmigaOS 3.1, 3.5 und 3.9 empfohlen.

1.4 Installation

Um Plananarama zu installieren, verwenden Sie einfach den mitgelieferten Installer oder kopieren Sie die Datei `plananarama.hwp` nach `LIBS:Hollywood`.

2 Über Plananarama

2.1 Danksagungen

Plananarama wurde von Andreas Falkenhahn geschrieben. Dieses Plugin wurde erstmals während der Entwicklung von Hollywood 6.0 als Proof-of-Concept für die Flexibilität der leistungsstarken Display- und Bitmap-Adapter von Hollywood 6.0 erstellt. Dieses Plugin ersetzt Hollywoods Standard-Display-Adapter vollständig und installiert einen benutzerdefinierten Adapter, der auf planaren Bildschirmen ausgeführt werden kann.

An Dominic Widmer und Helmut Haake geht ein besonderer Dank für die Übersetzung des Handbuchs ins Deutsche. Fehler oder Verbesserungsvorschläge bzgl. des deutschen Hollywood-Handbuchs bitte an das Übersetzungsteam richten, welches unter handbuch@gmx.ch erreicht werden kann.

Wenn Sie mich kontaktieren möchten, senden Sie bitte eine E-Mail an andreas@airsoftsoftwair.de oder nutzen Sie das Kontaktformular unter http://www.hollywood-mal.com.

2.2 Häufig gestellte Fragen

Dieser Abschnitt behandelt einige häufig gestellte Fragen. Bitte lesen Sie diese zuerst, bevor Sie auf der Mailingliste oder im Forum nachfragen, da Ihr Problem möglicherweise hier behandelt wurde.

F: Plananarama meldet "Cannot open guigfx.library!", aber ich habe die guigfx.library installiert.

A: guigfx.library erfordert render.library und kann nicht geöffnet werden, falls render.library nicht vorhanden ist. Stellen Sie also sicher, dass Sie auch render.library installiert haben. Wenn ja, dann überprüfen Sie, dass Sie die richtige Version der render.library haben. Einige erfordern eine FPU und können nicht geöffnet werden, wenn keine FPU vorhanden ist. Siehe Abschnitt 1.3 [Anforderungen], Seite 2, für Details.

F: Mein System stürzt mit einem 8000000B Softwarefehler ab.

A: Sie verwenden eine Version von render.library oder guigfx.library, die eine FPU benötigt, aber auf einem System ohne FPU läuft. Stellen Sie sicher, dass Sie die richtige Version für Ihr System verwenden, da einige Versionen von render.library und guigfx.library nicht prüfen, ob eine FPU vorhanden ist und somit stürzen sie einfach ab, wenn keine vorhanden ist. Siehe Abschnitt 1.3 [Anforderungen], Seite 2, für Details.

F: Ich erhalte die Fehlermeldung "Out of memory!", aber ich habe viel Fast-RAM.

A: Stellen Sie zunächst sicher, dass Sie FBlit installiert haben. Dann stellen Sie sicher, dass Sie FBlit richtig konfiguriert haben. Sie müssen Hollywood zur "Include"-Liste von FBlit auf der Registerkarte "FAllocBitMap" hinzufügen. Wenn Hollywood in dieser Liste enthalten ist, wird es in der Lage sein, Grafiken im Fast-RAM zu platzieren, was unbedingt benötigt

wird, da 2 MB Chip-RAM für Hollywood sicherlich nicht ausreichen werden. Stellen Sie also sicher, dass Sie FBlit richtig konfigurieren.

F: Plananarama ist sehr langsam.

A: Das kann viele Gründe haben. Wenn Sie den Tag `PaletteMode` in `@REQUIRE` auf `True` gesetzt haben, stellen Sie sicher, dass Sie den BlazeWCP-Patch aus dem Aminet installieren, um das Zeichnen zu beschleunigen. Im Paletten-Modus können Sie das Zeichnen auch erheblich beschleunigen, wenn Sie nur Palettengrafiken verwenden, d.h. keine Pinsel, BGPics, Animationen usw. verwenden, die als Hi-/True-Color-RGB-Bilder gespeichert sind, sondern nur Palettenbilder verwenden. Idealerweise sollten sie dieselbe Palette benutzen wie der Bildschirm, auf den Sie sie zeichnen möchten, damit Plananarama keine Neuzuordnung vornehmen muss, sondern sie einfach direkt zeichnen kann.

Ein weiterer Grund könnte sein, dass Ihr Skript viele Bilder mit Alphakanalgrafiken zeichnet, z.B. Text/Formen mit Anti-Aliasing oder Bilder mit unterschiedlichen Transparenzstufen. Diese Dinge machen auf palettenbasierten Bildschirmen nicht viel Sinn und sie belasten die CPU sehr stark, da Plananarama diese Bilder ständig von echtfarben, grobkörnigen Pixeln in planare Grafiken umwandeln muss.

Wenn Sie also Wert auf Leistung legen, sollten Sie keine Alphakanalbilder verwenden und die Transparenz auf monochrome Transparenz (d.h. sichtbare und unsichtbare Pixel) beschränken. Am besten verwenden Sie einen festen Satz vorgerenderter/vorberechneten Grafiken. Wenn Sie nur vorgerenderte Grafiken ohne Änderungen an den Farben zeichnen, sollte Plananarama auf planaren Bildschirmen recht gut funktionieren und Skripte sollten auch auf langsameren Systemen verwendbar sein, obwohl Sie natürlich immer noch eine schnelle CPU und etwas Fast-RAM benötigen.

F: Gibt es ein Hollywood-Forum, in dem ich mit anderen Benutzern in Kontakt treten kann?

A: Ja, bitte besuchen Sie die "Community" oder "Forum"-Sektion des offiziellen Hollywood-Portals unter `http://www.hollywood-mal.com`.

F: Wo kann ich um Hilfe bitten?

A: Es gibt ein lebhaftes englischsprachiges Forum auf `http://forums.hollywood-mal.com`. Ausserdem ist ein deutschsprachiges Forum vorhanden, welches Sie unter `https://www.amiga-resistance.info/` erreichen können. Sie können sich gerne beteiligen und dort Ihre Frage stellen.

F: Ich habe einen Fehler gefunden.

A: Bitte posten Sie darüber in der "Bugs"-Sektion des Forums.

2.3 Zukunft

Hier sind einige Punkte, die auf meiner Aufgabenliste stehen:

– Unterstützung für den Paletten-Modus auf öffentlichen Bildschirmen wie die Workbench hinzufügen; dies erfordert ein gemeinsames Stiftverwaltungssystem,

da Plananarama-Programme ihre Stifte mit anderen Programmen auf demselben Bildschirm teilen müssen

- Unterstützung für Hardware-Scrolling hinzufügen
- Unterstützung von Hardware-Doppelpuffern hinzufügen
- Unterstützung für Bobs hinzufügen

Zögern Sie nicht, mich zu kontaktieren, wenn Plananarama eine bestimmte Funktion fehlt, die für Ihr Projekt wichtig ist.

2.4 Geschichte

Bitte schauen Sie in die auf englisch verfasste Datei `history.txt`. Hier finden Sie ein vollständiges Änderungsprotokoll von Plananarama.

3 Verwendung

3.1 Erste Schritte

Nachdem Sie Plananarama installiert haben, wird es Hollywood automatisch verwenden, wenn CyberGraphX oder Picasso96 nicht verfügbar sind, weil Ihr System keine Grafikkarte hat. Nach der Installation werden also alle Ihre Hollywood-Skripte "automatisch" auf Palettenbildschirmen ausgeführt, da Hollywood sie durch Plananarama leitet. Wenn Sie alles Hollywood überlassen, werden Ihre Skripte im Remapping-Modus ausgeführt. Dieser Modus garantiert maximale Kompatibilität, benötigt aber auch viel Speicher und kann langsam sein, je nachdem, was das Skript macht.

Es ist auch möglich, Skripte zu schreiben, die speziell für Plananarama entwickelt wurden. Dadurch erzielen Sie in der Regel eine bessere Leistung, da Sie innerhalb der Einschränkungen von palettenbasierter Bildschirme arbeiten. Skripte, die speziell für Plananarama entwickelt wurden, sollten den Paletten-Modus von Plananarama anfordern, indem sie den Tag `PaletteMode` setzen, wenn sie das Plugin mit der Präprozessor-Anweisung `@REQUIRE` aufrufen, z.B:

```
@REQUIRE "plananarama", {PaletteMode = True}
```

Die obige Zeile weist Plananarama an, im Paletten-Modus zu arbeiten. Der Unterschied zwischen dem Paletten- und dem Remapping-Modus besteht darin, dass Ihr Skript im Paletten-Modus direkten Zugriff auf die Palettenstifte des Bildschirms hat. Wenn Sie beispielsweise den Befehl `SetPen()` von Hollywood im Paletten-Modus aufrufen, können Sie direkt die Farbe eines Bildschirmstiftes ändern und mit Hollywoods Befehl `SetPalette()` eine ganz neue Palette festlegen. Dies kann für verschiedene Effekte wie Farbverläufe oder Überblendungen verwendet werden, und da das Ändern der Bildschirmpalettenstifte von der benutzerdefinierten Chip-Hardware des Amiga verwaltet wird, ist das Ergebnis sofort und fast ohne Verzögerung sichtbar.

Da Sie im Paletten-Modus die volle Kontrolle über die Palette des Bildschirms haben, können Sie außerdem alle Ihre Grafiken so speichern, dass die Palette Ihrer Bilder mit der Palette des Bildschirms übereinstimmt. Wenn dies der Fall ist, können Grafiken sehr schnell gezeichnet werden, da keine Farbneuzuordnung durchgeführt werden muss und das Zeichnen von Grafiken nur eine Frage des Kopierens von Rohpixeln ist. Um also die beste Leistung mit Plananarama zu erzielen, sollten Sie das Plugin in den Paletten-Modus versetzen und dann Ihr Skript so gestalten, dass eine Farbneuzuordnung so weit wie möglich vermieden wird, z.B. indem alle Bilder dieselbe globale Palette verwenden. Das bedeutet auch, dass Sie beim Zeichnen von Grafikgrundelementen wie Rechtecken, Linien, Kreisen usw. keine RGB-Farben übergeben sollten, sondern stattdessen mit Palettenstiften zeichnen sollten. Dies ist möglich, indem Sie den Paletten-Modus wie folgt auf `#PALETTE_PEN` setzen:

```
SetPaletteMode(#PALETTEMODE_PEN)
```

Im Paletten-Modus können Sie auch Hardware-Sprites verwenden, was die Dinge weiter beschleunigen kann, da diese Sprites in kürzester Zeit gezeichnet werden können, weil sie vollständig von der Custom-Chip-Hardware des Amigas gehandhabt werden. Siehe Abschnitt 3.5 [Hardware-Sprites], Seite 14, für Details. Ein weiterer Vorteil der Verwendung des Paletten-Modus besteht darin, dass Ihr Skript guigfx.library und render.library nicht

benötigt. Um das Zeichnen im Paletten-Modus zu beschleunigen, wird jedoch empfohlen, BlazeWCP zu installieren. Siehe Abschnitt 1.3 [Anforderungen], Seite 2, für Details.

Beachten Sie, dass Plananarama im Paletten-Modus immer einen eigenen Bildschirm öffnet (selbst wenn das Hollywood-Skript ausdrücklich den Fenstermodus anfordert). Der Grund dafür ist, dass die volle Kontrolle über die Bildschirmpalette nur dann möglich ist, wenn Plananarama auf einem eigenen Bildschirm läuft. Bei der Ausführung auf der Workbench oder anderen Bildschirmen, die mit anderen Programmen geteilt werden, müssen auch die Palettenstifte geteilt werden, was die Sache komplizierter macht. Siehe Abschnitt 3.4 [Paletten-Modus], Seite 12, für Details.

Im Remapping-Modus, dem Standardmodus von Plananarama, kann Plananarama auf einem eigenen Bildschirm oder auf anderen Bildschirmen wie die Workbench ausgeführt werden. Falls der Remapping-Modus aktiv ist, gibt Ihnen Plananarama keinen Zugriff auf die Palettenstifte des Bildschirms, sondern passt die Farben aller gezeichneten Grafiken neu an, damit sie mit der Palette des Bildschirms übereinstimmen. Dies ist natürlich viel langsamer als das Zeichnen von Grafiken, deren Farben mit der Palette des Bildschirms im Paletten-Modus übereinstimmen (siehe oben), aber es ist sehr flexibel und ermöglicht es Ihnen, jedes Hollywood-Skript auf einem Palettenbildschirm auszuführen, solange Sie genügend freien Speicher haben. Siehe Abschnitt 3.3 [Remapping-Modus], Seite 12, für Details.

Beachten Sie, dass der Paletten-Modus Hollywood 9 oder besser benötigt. Der Remapping-Modus benötigt mindestens Hollywood 6.

3.2 Plananarama konfigurieren

Wenn sie mit der Präprozessor-Anweisung `@REQUIRE` das Plugin Plananarama deklarieren, können Sie die folgenden Tags übergeben:

`PaletteMode:`
: Mit diesem Tag kann festgelegt werden, ob Plananarama im Paletten-Modus ausgeführt werden soll oder nicht. Dies ist standardmäßig `False`, was den Remapping-Modus bedeutet (siehe oben für Details). (V2.0)

`NoBlackBackground:`
: Wenn dies auf `True` gesetzt ist, setzt Plananarama beim Öffnen im Vollbildmodus die Farbe 0 nicht auf Schwarz. Dies wird nur gehandhabt, wenn sich Plananarama im Remapping-Modus befindet. Im Paletten-Modus verwendet Plananarama die Palette des Displays. Die Voreinstellung ist `False`. (V2.0)

`SpriteResolution:`
: Dieser Tag kann verwendet werden, um eine bestimmte Sprite-Auflösung für die von Plananarama erstellten Hardware-Sprites zu erzwingen. Standardmäßig verwendet Plananarama die Sprite-Auflösung des Systems, die möglicherweise nicht Ihren Wünschen entspricht. Wenn z.B. die Sprite-Auflösung des Systems "Hires" ist, werden Ihre Sprites auch in "Hires" erscheinen, was möglicherweise nicht das ist, was Sie wollen. Die Sprite-Auflösung des Systems ist typischerweise identisch mit der Mauszeiger-Auflösung, die in den "Pointer"-Einstellungen des Systems eingestellt ist, weil AmigaOS den Mauszeiger unter Verwendung eines Hardware-Sprites implementiert. Wenn der Benutzer hier also einen Hires-Zeiger konfiguriert hat, verwenden alle Ihre Sprites standardmäßig auch Hires.

Wenn Sie das nicht möchten, setzen Sie diesen Tag auf 1, um die Auflösung von Lores-Sprites zu erzwingen. Um das Einstellen von Sprites zu erzwingen, setzen Sie den Tag auf 2. Dieser Tag ist standardmäßig auf 0 gesetzt, was bedeutet, dass die Sprite-Auflösung des Systems verwendet wird. Beachten Sie, dass dieser Tag nur auf AGA-Systemen nützlich ist, da Sprites auf ECS-Systemen immer Lores sind. (V2.0)

`PlanarOnly:`
Standardmäßig wird Plananarama nur gestartet, wenn der Bildschirm planar ist. Wenn Sie also `@REQUIRE` Plananarama auf RTG-Bildschirmen (15, 16, 24 und 32 Bit) verwenden, wird das Plugin überhaupt nicht starten, weil es normalerweise keinen Sinn macht, Plananarama auf RTG-Systemen zu verwenden. Es gibt jedoch eine Ausnahme: Wenn Sie einen benutzerdefinierten planaren Bildschirm mit Plananarama öffnen wollen, dann können Sie das auch auf RTG-Systemen tun, aber da Plananarama standardmäßig nicht auf RTG-Systemen startet, müssen Sie dieses Verhalten deaktivieren, indem Sie `PlanarOnly` auf `False` setzen. In diesem Fall wird Plananarama auch immer auf RTG-Systemen gestartet. Beachten Sie, dass wenn `PaletteMode` auf `True` gesetzt ist, `PlanarOnly` automatisch auf `False` gesetzt wird. Voreingestellt ist `True` (V2.1).

`DitherMode:`
Wenn sich Plananarama im Remapping-Modus befindet, kann dieser Tag verwendet werden, um den Dithering-Modus zu konfigurieren. Dieser kann auf `None`, `FS` (Standard), `Random` oder `Edd` eingestellt werden. Hier ist eine Beschreibung der verschiedenen Dither-Modi:

`None`	Kein Dithering.
`FS`	Floyd-Steinberg-Dithering. Dies ist die Voreinstellung.
`Random`	Zufälliges Dithering. Dieser Modus ist deutlich langsamer als das Floyd-Steinberg-Dithering.
`Edd`	EDD-Dithering. Dieser Modus ist schneller als das Floyd-Steinberg-Dithering.

Dieser Tag wird im Paletten-Modus ignoriert.

`DitherAmount:`
Wenn sich Plananarama im Remapping-Modus befindet, kann dieser Tag verwendet werden, um den Dither-Wert einzustellen. Dieser Wert muss zwischen 0 und 255 liegen. Derzeit ist dieser Wert nur für den `Random`-Dither-Modus sinnvoll. Der Standardwert ist 40. Dieser Tag wird im Paletten-Modus ignoriert.

`AutoDither:`
Wenn sich Plananarama im Remapping-Modus befindet, kann dieser Tag verwendet werden, um automatisches Dithering zu aktivieren. Wenn diese Option auf `True` gesetzt ist, wird Dithering automatisch aktiviert, um ein bestimmtes Bild in eine bestimmte Umgebung zu zeichnen, wenn der Verlust von Farbinformationen einen bestimmten Schwellenwert überschreiten würde (siehe unten). Die Voreinstellung ist `True`. Dieser Tag wird im Paletten-Modus ignoriert.

`DitherThreshold:`
: Wenn sich Plananarama im Remapping-Modus befindet, kann diese Option verwendet werden, um den Schwellenwert für das automatische Dithering festzulegen. Je niedriger, desto früher wird das automatische Dithering aktiviert. Nützliche Schwellenwerte liegen zwischen 10 und 10000. Weitere Einzelheiten finden Sie in render.library/RGBArrayDiversityA(). Verwenden Sie diesen Tag besser nicht, es sei denn, Sie haben einen guten Grund dazu. Überlassen Sie es dem Benutzer, es mit der Umgebungsvariable `AUTODITHERTHRESHOLD` anzupassen. Die Voreinstellung ist 250. Dieser Tag wird im Paletten-Modus ignoriert.

`Precision:`
: Wenn sich Plananarama im Remapping-Modus befindet, kann dieselbe Option verwendet werden, um die Genauigkeit für die Stiftzuweisungen einzustellen. Dies kann `Exact`, `Image` (Voreinstellung), `Icon` oder `GUI` sein. Einzelheiten finden Sie unter graphics.library/ObtainBestPenA(). Beachten Sie, dass die Standardpräzision für fast jede Anwendung ausreicht. Stifte werden auf äußerst effektive Weise ermittelt. Auch bei geringeren Genauigkeiten erzielen Sie hervorragende Ergebnisse. Commodores Idee mit ObtainBestPenA() war es, einen ordentlichen und effektiven Mechanismus zur gemeinsamen Nutzung von Stiften zu schaffen, und Plananarama verhält sich in perfekter Übereinstimmung mit dieser Intention. Verwenden Sie niemals irgendwelche Patches für ObtainBestPenA(). Dieser Tag wird im Paletten-Modus ignoriert.

3.3 Remapping-Modus

Standardmäßig läuft Plananarama im Remapping-Modus. Der Remapping-Modus ist der praktischste der Darstellungsmodi von Plananarama, da Sie damit einfach jedes Hollywood-Skript auf allen Arten von Palettenbildschirmen ausführen können. Sie müssen Ihren Code in keiner Weise anpassen, er funktioniert einfach, weil alle Grafiken an die Palette des Zielbildschirms angepasst werden. Dies bedeutet jedoch viel Arbeit für die CPU, sodass es sehr langsam sein wird. Außerdem bedeutet dies, dass alle Bilder als RGB-Grafiken gespeichert werden, was viel Speicherplatz verbraucht. Beispielsweise benötigt ein 640x480-Bild 1,2 MB Speicherplatz, wenn es als RGB gespeichert wird, aber nur 300kB, wenn es hingegen als Palettenbild gespeichert wird.

Um die Dinge zu beschleunigen, könnten Sie stattdessen versuchen, den Paletten-Modus zu verwenden, aber dazu müssen Sie Ihr Skript speziell an die Einschränkungen eines palettenbasierten Displays anpassen. Siehe Abschnitt 3.4 [Paletten-Modus], Seite 12, für Details.

Beachten Sie, dass bei Verwendung von Plananarama im Remapping-Modus die Installation von FBlit empfohlen wird. Siehe Abschnitt 1.3 [Anforderungen], Seite 2, für Details.

3.4 Paletten-Modus

Wenn Sie den Paletten-Modus mit Plananarama verwenden, können Sie eine bessere Leistung erzielen als im Remapping-Modus, aber das hat den Preis, dass Sie Ihr Skript speziell für den Paletten-Modus von Plananarama entwerfen müssen. Beispielsweise müssen Sie sicherstellen, dass Ihr Hollywood-Display ein Paletten-Display ist. Sonst bekommt man natürlich keine Leistungsverbesserung, denn wenn das Hollywood-Display keine Palette,

sondern RGB-Grafiken verwendet, müssen trotzdem alle Grafiken neu zugeordnet werden, genau wie im Remapping-Modus.

Hier ist ein Beispielcode, der ein Paletten-Display einrichtet und Plananarama in den Paletten-Modus versetzt:

```
@REQUIRE "plananarama", {PaletteMode = True}
@DISPLAY {Palette = #PALETTE_AGA}

SetPaletteMode(#PALETTEMODE_PEN)
SetFillStyle(#FILLCOLOR)
SetDrawPen(2)
Box(#CENTER, #CENTER, 320, 240)
```

Der Code führt mehrere sehr wichtige Schritte aus, die notwendig sind, um den Paletten-Modus von Plananarama voll auszunutzen: Zuerst erstellt er ein Paletten-Display, indem er den Tag `Palette` verwendet, um dem Display die integrierte Palette `#PALETTE_AGA` zuzuweisen. Alternativ könnten Sie auch ein Paletten-Display erstellen, indem Sie ihr einfach ein Paletten-BGPic zuweisen, z.B. so:

```
@REQUIRE "plananarama", {PaletteMode = True}
@BGPIC 1, "background.iff", {LoadPalette = True}
```

Da wir `LoadPalette` im obigen Code auf `True` setzen, wird Ihr Display automatisch zu einem Paletten-Display, da ihr BGPic ein Palettenbild ist.

Die zweite sehr wichtige Sache, die der erste Codeausschnitt macht, ist das Aufrufen von `SetPaletteMode()` mit `#PALETTEMODE_PEN`, das ihm übergeben wird. Dies ist sehr wichtig, denn wenn Sie dies nicht tun, wird Hollywood immer noch alle Grafiken auf die Palette Ihres Displays neu zuordnen, was langsam ist. Nur wenn Sie den Paletten-Modus auf `#PALETTEMODE_PEN` setzen, können Sie Hollywood anweisen, keine Neuzuordnung vorzunehmen, sondern nur die Rohpixel zu kopieren. Dies bedeutet natürlich, dass beim Zeichnen von Bildern deren Palette mit der Display-Palette übereinstimmen muss, da Sie sonst falsche Farben erhalten.

Schließlich ruft der Codeausschnitt `SetDrawPen()` auf, um einen Zeichenstift festzulegen. Dieser Schritt ist sehr wichtig, wenn Sie Grafikgrundelemente wie Linien, Rechtecke, Kreise usw. zeichnen möchten. Wenn der Paletten-Modus auf `#PALETTEMODE_PEN` gesetzt wurde, ignorieren Hollywood-Befehle wie `Box()`, `Line()`, `Circle()` usw. die ihnen übergebene RGB-Farbe. Stattdessen zeichnen sie mit dem Stift, der mit `SetDrawPen()` eingestellt wurde. Aus diesem Grund zeichnet der obige Code ein weißes Rechteck und kein schwarzes, obwohl das Farb-Argument im Aufruf von `Box()` standardmäßig schwarz ist, weil es weggelassen wurde.

Da wir die volle Kontrolle über die Hardware-Farbregister haben, könnten wir das weiße Rechteck jetzt leicht in ein rotes umwandeln, indem wir einfach die Farbe von Palettenstift 2 ändern. Dies kann wie folgt geschehen:

```
SetPen(2, #RED)
```

Dann könnten wir das rote Rechteck sanft zu Schwarz ausblenden, indem wir etwas wie folgt tun:

```
For Local k = 32 To 0 Step -1
   SetPen(2, RGB(255 * (k/32), 0, 0))
```

```
        VWait
    Next
```

Natürlich könnten Sie auch die Palettenfarben durchlaufen und mit Hollywoods Befehl `SetPalette()` eine komplett neue Palette anwenden. Im Paletten-Modus sind viele Dinge möglich.

Ein weiterer Vorteil der Verwendung des Paletten-Modus besteht darin, dass Ihr Skript guigfx.library und render.library nicht benötigt. Um das Zeichnen im Paletten-Modus zu beschleunigen, wird jedoch empfohlen, BlazeWCP zu installieren. Siehe Abschnitt 1.3 [Anforderungen], Seite 2, für Details.

3.5 Hardware-Sprites

Ab Plananarama 2.0 unterstützt das Plugin auch Amiga-Hardware-Sprites. Da diese von der benutzerdefinierten Chipsatz-Hardware des Amiga verwaltet werden, können sie ohne Leistungseinbußen äußerst effizient gezeichnet werden. Allerdings waren Sprites schon immer die Achillesferse des Amigas, da sie im Vergleich zu anderen Systemen (insbesondere im Vergleich zu Spielkonsolen) recht eingeschränkt sind.

Konkret gibt es die folgenden Einschränkungen, wenn es um Hardware-Sprites auf dem Amiga geht:

- Es gibt nur 8 Sprite-DMA-Kanäle, so dass Sie nur maximal 8 Sprites haben können
- Jeder Sprite-DMA-Kanal kann nur 4 Farbgrafiken verarbeiten
- Glücklicherweise können zwei Sprite-DMA-Kanäle kombiniert werden, um ein 16-Farben-Sprite zu erstellen; aber wenn Sie 16-Farben-Sprites verwenden, können Sie nur 4 davon haben, weil jedes 16-Farben-Sprite zwei Sprite-DMA-Kanäle belegt
- Auf OCS/ECS beträgt die maximale Sprite-Breite 16 Pixel
- Bei AGA beträgt die maximale Sprite-Breite 64 Pixel
- Es gibt keine maximale Spritehöhe
- Die einzelnen Sprite-DMA-Kanäle sind an bestimmte Farbregister gebunden, und eine Farbe ist immer für die Transparenz reserviert. Siehe Abschnitt 4.1 [planar.CreateSprite], Seite 17, für Details.

Aufgrund all dieser Einschränkungen werden Sie mit Amiga-Hardware-Sprites keine Berge versetzen können, aber wenn Sie nur ein paar Sprites benötigen, können sie trotzdem recht nützlich sein, weil sie so schnell gezeichnet werden können, da sie vollständig auf der Hardware gehandhabt werden.

Beachten Sie, dass Sie bei der Verwendung von Hardware-Sprites Plananarama im Paletten-Modus verwenden sollten, da Sie im Remapping-Modus keine Kontrolle über die Palettenstifte des Bildschirms haben, sodass es keine Möglichkeit gibt, die Sprite-Farben einzustellen. Siehe Abschnitt 3.4 [Paletten-Modus], Seite 12, für Details.

Beachten Sie auch, dass Sie möglicherweise den Tag `SpriteResolution` setzen sollten, wenn Sie Hardware-Sprites verwenden. Andernfalls verwenden Ihre Sprites die Sprite-Auflösung des Systems, was möglicherweise nicht Ihren Wünschen entspricht. Z.B. wenn die Sprite-Auflösung des Systems "Hires" ist, werden Ihre Sprites auch in "Hires" erscheinen, was möglicherweise nicht das ist, was Sie wollen. Die Sprite-Auflösung des Systems ist typischerweise identisch mit der Mauszeiger-Auflösung, die in den

"Pointer"-Einstellungen des Systems eingestellt ist, weil AmigaOS den Mauszeiger unter Verwendung eines Hardware-Sprites implementiert. Wenn der Benutzer hier also einen Hires-Zeiger konfiguriert hat, verwenden alle Ihre Sprites standardmäßig auch "Hires". Wenn Sie das nicht möchten, setzen Sie den Tag `SpriteResolution` auf 1, um Lores-Sprites zu erzwingen. Siehe Abschnitt 3.2 [Plananarama konfigurieren], Seite 10, für Details. Beachten Sie, dass der Tag `SpriteResolution` nur auf AGA-Systemen wirklich benötigt wird, da Sprites auf ECS-Systemen immer Lores sind.

3.6 RapaGUI und MUI Royale

Plananarama unterstützt auch die Plugins RapaGUI und MUI Royale. Wenn Plananarama installiert ist, werden sowohl RapaGUI als auch MUI Royale automatisch auf Palettenbildschirmen ausgeführt. Beachten Sie jedoch, dass Sie den Paletten-Modus von Plananarama nicht verwenden dürfen, wenn Sie RapaGUI und MUI Royale verwenden. Mit diesen Plugins muss Plananarama immer im Remapping-Modus verwendet werden.

4 Befehlsreferenz

4.1 planar.CreateSprite

BEZEICHNUNG

planar.CreateSprite – erstellt Hardware-Sprites aus einem Pinsel (V2.0)

ÜBERSICHT

```
[id] = planar.CreateSprite(id, brushid)
```

BESCHREIBUNG

Dieser Befehl wandelt den durch `brushid` angegebenen Pinsel in ein Hardware-Sprite um und weist ihm den Identifikator `id` zu. Wenn Sie im Argument `id Nil` angeben, wählt `planar.CreateSprite()` automatisch einen freien Identifikator für diesen Sprite und gibt ihn zurück.

Beachten Sie, dass der Pinsel, den Sie an diesen Befehl übergeben, die Sprite-Einschränkungen der Amiga-Hardware respektieren muss. Das bedeutet, dass er sich an die folgenden Regeln halten muss:

- Es muss ein Palettenpinsel sein
- Auf OCS/ECS-Systemen beträgt die maximale Sprite-Breite 16 Pixel
- Auf AGA-Systemen beträgt die maximale Sprite-Breite 64 Pixel
- Der Palettenpinsel muss entweder 4 oder 16 Farben verwenden

Beachten Sie auch, dass die Amiga-Hardware nur 8 Sprite-DMA-Kanäle unterstützt. Jeder Kanal kann ein 4-Farben-Sprite haben. Die 8 Sprite-DMA-Kanäle sind mit den folgenden Farbregistern verbunden:

- Kanäle 0 und 1: Farbregister 16 bis 19 (Farbe 16 ist transparent). Beachten Sie, dass Kanal 0 für die Verwendung durch Intuition für den Mauszeiger reserviert ist.
- Kanäle 2 und 3: Farbregister 20 bis 23 (Farbe 20 ist transparent).
- Kanäle 4 und 5: Farbregister 20 bis 23 (Farbe 24 ist transparent).
- Kanäle 6 und 7: Farbregister 20 bis 23 (Farbe 28 ist transparent).

Zwei Kanäle können kombiniert werden, um ein Sprite mit 16 Farben zu erstellen. Das bedeutet, wenn Sie Sprites mit 16 Farben verwenden, können Sie nur 4 statt 8 haben, da ein Sprite mit 16 Farben zwei Sprite-DMA-Kanäle blockiert. 16 Farbsprites sind an die Farbregister 16 bis 31 gebunden (Farbe 16 ist transparent).

Beachten Sie, dass `planar.CreateSprite()` den Sprite nicht sofort einem Sprite-DMA-Kanal zuordnet. Dies ist die Aufgabe von planar.MapSprite(). Somit können Sie mit `planar.CreateSprite()` mehr Hardware-Sprites erstellen, als Sprite-DMA-Kanäle vorhanden sind. Dies ist beispielsweise nützlich, wenn Sie Sprites animieren möchten. In diesem Fall könnten Sie zuerst alle Animationsframes mit `planar.CreateSprite()` in Hardware-Sprites konvertieren und dann die einzelnen Animationsframes vor/nach dem Anzeigen mit planar.MoveSprite() mappen und unmapen.

EINGABEN

`id`	Identifikator für den Hardware-Sprite oder `Nil` für die automatische ID-Auswahl

`brushid` Identifikator des Palettenpinsels, der in ein Hardware-Sprite konvertiert werden soll

RÜCKGABEWERTE

`id` optional: Identifikator des Hardware-Sprites; wird nur zurückgegeben, wenn Sie `Nil` als Argument 1 übergeben (siehe oben)

BEISPIEL

```
@REQUIRE "plananarama", {PaletteMode = True}
@DISPLAY {Palette = #PALETTE_AGA}
SetPaletteMode(#PALETTEMODE_PEN)
SetFillStyle(#FILLCOLOR)
CreateBrush(1, 64, 64, {Palette = #PALETTE_GRAY4})
SelectBrush(1)
For Local k = 0 To 2
   SetDrawPen(k + 1)
   Box(k * 21, 0, 21, 64)
Next
EndSelect
planar.CreateSprite(1, 1)
planar.MapSprite(1)
Repeat
   planar.MoveSprite(1, MouseX(), MouseY())
   planar.VWait()
Forever
```

Der obige Code erstellt ein 4-Farben-64x64-Sprite, ordnet ihn einem Sprite-DMA-Kanal zu und bewegt ihn dann dorthin, wo sich der Mauszeiger befindet. Beachten Sie, dass es keine Rolle spielt, dass wir `#PALETTE_GRAY4` an `CreateBrush()` übergeben, da der Hardware-Sprite die Palette des Bildschirms verwendet, also verwenden wir einfach `#PALETTE_GRAY4` als Dummy, um `CreateBrush()` mitzuteilen, dass es uns ein 16-Farben-Sprite geben soll.

4.2 planar.FreeSprite

BEZEICHNUNG

planar.FreeSprite – löscht einen Hardware-Sprite aus dem Speicher (V2.0)

ÜBERSICHT

```
planar.FreeSprite(id)
```

BESCHREIBUNG

Dieser Befehl löscht den durch `id` angegebenen Hardware-Sprite aus dem Speicher. Wenn der Sprite derzeit einem Sprite-DMA-Kanal zugeordnet ist, wird er vor dem Löschen aufgehoben.

EINGABEN

`id` ID des Hardware-Sprites, um es zu löschen

4.3 planar.GetSpriteType

BEZEICHNUNG

planar.GetSpriteType – holt sich den Objekttyp des Hardware-Sprites (V2.0)

ÜBERSICHT

```
type = planar.GetSpriteType()
```

BESCHREIBUNG

Dieser Befehl gibt den von Plananarama registrierten Hardware-Sprite-Objekttyp zurück. Sie können diesen Objekttyp dann an Hollywoods Befehl `GetAttribute()` übergeben, um die folgenden Attribute von Hardware-Sprites abzufragen:

`#ATTRWIDTH:`
: Die Hardware-Sprite-Breite.

`#ATTRHEIGHT:`
: Die Hardware-Sprite-Höhe.

`#ATTRDEPTH:`
: Die Hardware-Sprite-Tiefe.

`#ATTRXPOS:`
: Die X-Position des Hardware-Sprites.

`#ATTRYPOS:`
: Die Y-Position des Hardware-Sprites.

`#ATTRSTATE:`
: Der Sprite-DMA-Kanal, dem der Hardware-Sprite zugeordnet wurde. Für nicht zugeordnete Hardware-Sprites ist dies -1.

EINGABEN

keine

RÜCKGABEWERTE

`type` von Plananarama registrierter Hardware-Sprite-Objekttyp

BEISPIEL

```
DMASPRITE_TYPE = planar.GetSpriteType()
w = GetAttribute(DMASPRITE_TYPE, 1, #ATTRWIDTH)
h = GetAttribute(DMASPRITE_TYPE, 1, #ATTRHEIGHT)
```

Der obige Code fragt die Breite und Höhe von Hardware-Sprite 1 ab.

4.4 planar.HaveAGA

BEZEICHNUNG

planar.HaveAGA – prüft, ob ein AGA-Chipsatz vorhanden ist (V2.0)

ÜBERSICHT

```
ok = planar.HaveAGA()
```

BESCHREIBUNG

Dieser Befehl gibt `True` zurück, wenn der AGA-Chipsatz vorhanden ist, andernfalls `False`.

EINGABEN

keine

RÜCKGABEWERTE

`ok` `True` oder `False`, um anzugeben, ob der AGA-Chipsatz vorhanden ist oder nicht

4.5 planar.MapSprite

BEZEICHNUNG

planar.MapSprite – ordnet den Hardware-Sprite einem DMA-Kanal zu (V2.0)

ÜBERSICHT

```
planar.MapSprite(id[, t])
```

BESCHREIBUNG

Dieser Befehl ordnet den durch `id` angegebenen Hardware-Sprite einem freien Sprite-DMA-Kanal zu. Der von `id` angegebene Hardware-Sprite muss mit planar.CreateSprite() erstellt worden sein.

Weitere Parameter können im optionalen Tabellenargument angegeben werden. Die folgenden Tags sind derzeit erkannt:

`Channel:` Standardmäßig wählt `planar.MapSprite()` einen Sprite-DMA-Kanal automatisch aus. Wenn Sie einen bestimmten Sprite-DMA-Kanal verwenden möchten, können Sie ihn mit diesem Tag einstellen. Dies muss eine Zahl zwischen 1 und 7 sein (Kanal 0 ist durch Intuition für den Mauszeiger-Sprite reserviert). Wenn der Sprite 16 Farben verwendet, können Sie nur die Kanäle 2, 4 oder 6 verwenden, da 16 Farbsprites zwei benachbarte Sprite-DMA-Kanäle einnehmen.

`Display:` Dies kann auf die ID eines Hollywood-Displays eingestellt werden, auf dem der Sprite erscheinen soll. Dies ist normalerweise nicht erforderlich da der Sprite einfach das aktuelle Hollywood-Display verwendet.

Beachten Sie, dass Sie, wenn der Sprite-DMA-Kanal automatisch von `planar.MapSprite()` ausgewählt wird, das mit dem Attribut `#ATTRSTATE` abfragem können, um den DMA-Kanal herauszufinden, dem Ihr Sprite zugeordnet wurde, nachdem Sie `planar.MapSprite()` aufgerufen haben.

Verwenden Sie zum Aufheben der Zuordnung eines Hardware-Sprites von einem DMA-Kanal planar.UnmapSprite(). Siehe Abschnitt 4.7 [planar.UnmapSprite], Seite 21, für Details.

EINGABEN

`id` ID des Hardware-Sprites, der dem DMA-Kanal zugeordnet werden soll

`t` Optional: Tabelle mit weiteren Argumenten (siehe oben)

BEISPIEL
Siehe Abschnitt 4.1 [planar.CreateSprite], Seite 17.

4.6 planar.MoveSprite

BEZEICHNUNG
planar.MoveSprite – legt die Sprite-Position fest (V2.0)

ÜBERSICHT
`planar.MoveSprite(id, x, y)`

BESCHREIBUNG
Dieser Befehl bewegt den in `id` angegebenen Hardware-Sprite an die von `x` und `Y` angegebene Position. Dies ist nur für Sprits möglich, die einem Sprite-DMA-Kanal zugeordnet wurden, bevor Sie den Befehl planar.MapSprite() verwenden.

EINGABEN

`id`	Identifikator des zu verschiebenden Hardware-Sprites
`x`	gewünschte neue x-Position
`y`	gewünschte neue y-Position

BEISPIEL
```
see @link{planarCreateSprite, planar.CreateSprite()}
```

4.7 planar.UnmapSprite

BEZEICHNUNG
planar.UnmapSprite – hebt die Zuordnung des Hardware-Sprites zum DMA-Kanal auf (V2.0)

ÜBERSICHT
`planar.UnmapSprite(id)`

BESCHREIBUNG
Dieser Befehl hebt die Zuordnung des durch `id` angegebenen Hardware-Sprite vom Sprite-DMA-Kanal auf, zu dem er derzeit gehört. Der Sprite muss einem DMA-Kanal mit dem Befehl planar.MapSprite() zuvor zugeordnet worden sein.

EINGABEN

`id`	Identifikator des Hardware-Sprites, dessen Zuordnung vom DMA-Kanal aufgehoben wird

4.8 planar.VWait

BEZEICHNUNG
planar.VWait – wartet auf den vertikalen leeren Interrupt (V2.0)

ÜBERSICHT

`planar.VWait()`

BESCHREIBUNG

Dieser Befehl wartet auf den vertikalen leeren Interrupt. Bei der Verwendung von Plananarama ist es besser, diesen Befehl anstelle von Hollywoods eigenen `Vwait()` zu verwenden, da der Befehl planar.vwait() auf einem Niveau arbeitet, das an der benutzerdefinierten Chip-Hardware dem Amiga näher ist.

EINGABEN

keine

Index