RebelSDL 1.1

"I'm just a rebel"

Andreas Falkenhahn

Inhaltsverzeichnis

1 Allgemeine Information 1

1.1 Einführung 1
1.2 Lizenzvereinbarung 2
1.3 Anforderungen 3
1.4 Installation 3

2 Über RebelSDL 5

2.1 Danksagungen 5
2.2 Häufig gestellte Fragen 5
2.3 Bekannte Probleme 6
2.4 Zukunft 7
2.5 Geschichte 7

3 Verwendung 9

3.1 RebelSDL aktivieren 9
3.2 Verwendung eines Hardware-Doppelpuffers 11
3.3 Zeichnen von Grafiken 11
3.4 Verwendung von Hardwarepinseln 12
3.5 Offscreen-Wiedergabe 13
3.6 Verwendung der SDL-Wiedergabe 14
3.7 Joysticks und Gamecontroller 14
3.8 Erhöhung der Ausführungsgeschwindigkeit 14
3.9 RebelSDL als Hilfs-Plugin 15
3.10 Raspberry Pi Besonderheiten 16

4 Beispiele 17

4.1 Beispiele 17

5 Joystick-Referenz 19

5.1 sdl.ForceJoystickMode 19
5.2 sdl.GetAxis 19
5.3 sdl.GetBall 20
5.4 sdl.GetButton 20
5.5 sdl.GetHat 21
5.6 sdl.GetJoysticks 22
5.7 sdl.GetNumAxes 22
5.8 sdl.GetNumBalls 22
5.9 sdl.GetNumButtons 23
5.10 sdl.GetNumHats 23
5.11 sdl.IsGameController 24

6 Tastatur-Referenz ... 25

6.1 sdl.SetTextInputRect ... 25
6.2 sdl.StartTextInput ... 25
6.3 sdl.StopTextInput ... 25

7 Wiedergabe-Referenz ... 27

7.1 sdl.EnableOffscreenRender ... 27
7.2 sdl.GetCurrentRenderDriver ... 27
7.3 sdl.GetRenderDrawBlendMode ... 28
7.4 sdl.GetRenderDrawColor ... 28
7.5 sdl.GetRendererOutputSize ... 29
7.6 sdl.GetTextureAlphaMod ... 29
7.7 sdl.GetTextureBlendMode ... 29
7.8 sdl.GetTextureColorMod ... 30
7.9 sdl.RenderClear ... 30
7.10 sdl.RenderCopy ... 31
7.11 sdl.RenderDrawLine ... 32
7.12 sdl.RenderDrawPoint ... 32
7.13 sdl.RenderDrawRect ... 33
7.14 sdl.RenderFillRect ... 33
7.15 sdl.RenderGetClipRect ... 34
7.16 sdl.RenderGetLogicalSize ... 34
7.17 sdl.RenderGetScale ... 35
7.18 sdl.RenderGetViewport ... 35
7.19 sdl.RenderPresent ... 36
7.20 sdl.RenderSetClipRect ... 36
7.21 sdl.RenderSetLogicalSize ... 37
7.22 sdl.RenderSetScale ... 37
7.23 sdl.RenderSetViewport ... 38
7.24 sdl.SetRenderDrawBlendMode ... 38
7.25 sdl.SetRenderDrawColor ... 39
7.26 sdl.SetRenderTarget ... 39
7.27 sdl.SetScaleQuality ... 40
7.28 sdl.SetTextureAlphaMod ... 40
7.29 sdl.SetTextureBlendMode ... 41
7.30 sdl.SetTextureColorMod ... 42

8 System-Referenz ... 43

8.1 sdl.ClearError ... 43
8.2 sdl.GetCurrentVideoDriver ... 43
8.3 sdl.GetError ... 43
8.4 sdl.GetVersion ... 44

9 Fenster-Referenz ... 45

9.1 sdl.SetWindowFullscreen ... 45

Anhang A Lizenzen **47**
A.1 SDL-Lizenzen 47

Index **49**

1 Allgemeine Information

1.1 Einführung

RebelSDL ist ein Plugin für Hollywood, mit dem Sie SDL (Simple DirectMedia Layer) von Hollywood verwenden können. Dies ermöglicht es, Skripte zu schreiben, die die Grafikhardware des Hostsystems verwenden. Damit entwickeln Sie eine leistungsstarke butterweiche 2D-Animation, die vollständig in der GPU-Hardware Ihrer Grafikkarte erstellt wird. Dies führt zu einer enormen Leistungssteigerung gegenüber der klassischen Hollywood-Grafik-API, die größtenteils in der Software für maximale Portabilität und Kompatibilität implementiert ist. Besonders Systeme mit langsameren CPUs (wie der Raspberry Pi) werden von der hardwarebasierten Zeichnung, Skalierung und Transformation von SDL profitieren.

SDL ist eine plattformübergreifende Entwicklungsbibliothek, die über OpenGL und Direct3D einen Low-Level-Zugriff auf Audio-, Tastatur-, Maus-, Joystick- und Grafikhardware ermöglicht. Es wird von Videowiedergabesoftware, Emulatoren und beliebten Spielen verwendet. Weitere Informationen zu SDL erhalten Sie unter `http://www.libsdl.org`. Zum Erlernen von SDL können Sie gute Anleitungen im gesamten Web finden.

RebelSDL ersetzt Hollywoods integrierten Display-Handler transparent durch einen eigenen, von SDL verwalteten Display-Handler. Wenn RebelSDL aktiviert ist, werden Hollywood-Displays automatisch SDL-Fenstern zugeordnet und Hardwarepinsel werden direkt auf SDL-Texturen abgebildet, so dass sie auf allen unterstützten Systemen extrem schnell gezeichnet, skaliert und transformiert werden können. Dies ist besonders nützlich unter Windows, macOS und Linux, da Hollywood standardmäßig keine Hardware-Doppelpuffer und Hardwarepinsel auf diesen Plattformen unterstützt. Mit RebelSDL können jetzt aber auch Hardware-Doppelpuffer und Hardwarepinsel auf diesen Plattformen eingesetzt werden. So kann RebelSDL hier auch als Helfer-Plugin fungieren, welches diese Funktionalität zu Hollywood hinzufügt, ohne dass Sie eine einzelne Zeile SDL-Code schreiben müssen!

Darüber hinaus bietet RebelSDL Schnittstellen-Funktionen für einige nützliche Befehle der SDL-API. RebelSDL ermöglicht beispielsweise den Aufruf von Joystick- und Gamecontroller-Funktionen in SDL, die viel flexibler sind als die eingebaute Joystick-Bibliothek von Hollywood. Mit RebelSDL können Sie außerdem auf Hollywood-Hardwarepinsel als SDL-Texturen zugreifen und deren Eigenschaften über einige von RebelSDL bereitgestellte dedizierte SDL-Aufrufe ändern.

Schließlich ersetzt RebelSDL auch Hollywoods eingebauten Audiotreiber mit dem von SDL angebotenen Audiotreiber. Im Gegensatz zum Grafiktreiber hat SDLs Audiotreiber wahrscheinlich keinen Vorteil gegenüber Hollywoods eingebautem Audiotreiber, aber durch die Verwendung von RebelSDL wird Ihr Programm zu einer vollständigen SDL-Anwendung, die SDL nicht nur für die Grafikausgabe, sondern auch für die Audioausgabe verwendet.

RebelSDL verwendet die neue Plugin-Schnittstelle für den Display-Adapter, die mit Hollywood 6.0 eingeführt wurde. Daher funktioniert das Plugin nicht mit älteren Versionen von Hollywood. Es erfordert mindestens Hollywood 6.0. Wenn RebelSDL aktiviert ist, werden alle Grafik- und Audioausgaben automatisch über SDL weitergeleitet. Um vom hardwarebeschleunigten Zeichnen zu profitieren, müssen Hollywood-Skripte jedoch einigen Regeln folgen, die in diesem Handbuch beschrieben sind.

RebelSDL enthält umfangreiche Dokumentationen in verschiedenen Formaten wie PDF, HTML, AmigaGuide und CHM, die Informationen zur Verwendung dieses Plugins enthalten. Darüber hinaus sind viele Beispielskripte im Distributionarchiv enthalten, um wirklich schnell loslegen zu können.

All dies macht RebelSDL zum ultimativen Scripting-Erlebnis für alle SDL-Rebellen, indem es das Beste aus beiden Welten in einem leistungsstarken Plug-in vereint: Hollywoods umfangreiches und praktisches Multimediafunktionsset und SDLs rohe Grafikleistung!

1.2 Lizenzvereinbarung

Das Programm wird zur Verfügung gestellt "wie es ist" und der Autor kann für keinerlei Schäden, welcher Natur sie auch immer sein mögen, verantwortlich gemacht werden. Sie benutzen dieses Programm völlig auf eigene Gefahr und eigenes Risiko. Der Autor gibt keinerlei Garantien in Verbindung mit der Benutzung dieses Programmes, nicht einmal die Garantie der Funktionstüchtigkeit.

RebelSDL kann frei weitergegeben werden solange die folgenden drei Bedingungen erfüllt sind:

1. Es dürfen keine Änderungen am Programm vorgenommen werden.
2. Das Programm darf nicht verkauft werden.
3. Wenn Sie das Programm auf einer Coverdisk veröffentlichen möchten, müssen Sie erst um Erlaubnis fragen.

Dieses Programm benutzt den Simple DirectMedia Layer (SDL) Copyright (C) 1997-2016 Sam Lantinga. Siehe Abschnitt A.1 [SDL-Lizenzen], Seite 47, für Details.

Dieses Handbuch basiert in Teilen auf dem SDL-Handbuch, welches hier eingesehen werden kann: `http://wiki.libsdl.org/FrontPage`

Alle Warenzeichen sind Eigentum ihrer jeweiligen Firmen.

FÜR DIESES PROGRAMM GIBT ES KEINE GARANTIE, SOWEIT ES DIE ANZUWENDENDEN GESETZE ZULASSEN. SOFERN ANDERSWO NICHTS GEGENTEILIGES GESCHRIEBEN STEHT STELLEN DER AUTOR UND/ODER DRITTE DAS PROGRAMM "SO WIE ES IST" ZUR VERFÜGUNG, OHNE IRGENDEINE GARANTIE, WEDER DIREKT NOCH INDIREKT. DIES BEINHALTET, IST ABER NICHT DARAUF BESCHRÄNKT, VERKÄUFLICHKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN VERWENDUNGSZWECK. DAS VOLLSTÄNDIGE RISIKO DER QUALITÄT UND AUSFÜHRBARKEIT DES PROGRAMMS LIEGT BEIM ANWENDER. SOLLTE SICH DAS PROGRAMM ALS DEFEKT HERAUSSTELLEN, LIEGEN ALLE KOSTEN FÜR SERVICE, INSTANDSETZUNG ODER NACHBESSERUNG BEIM ANWENDER.

KEIN COPYRIGHT-INHABER ODER DRITTER, DER DAS PROGRAMM WIE OBEN ERLAUBT WEITERVERKAUFT, KANN FÜR SCHÄDEN IRGENDWELCHER ART HAFTBAR GEMACHT WERDEN (DIES BEINHALTET, IST ABER NICHT BESCHRÄNKT AUF, DATENVERLUST INFOLGE UNFÄHIGKEIT DES PROGRAMMS, MIT ANDEREN PROGRAMMEN ZUSAMMENZUARBEITEN), SELBST WENN EIN

SOLCHER INHABER ODER DRITTER AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE, AUSSER ES BESTEHT EINE SCHRIFTLICHE EINWILLIGUNG ODER WIRD VOM GESETZ VERLANGT.

1.3 Anforderungen

- Hollywood 6.0 oder besser
- Windows: erfordert mindestens Windows 2000
- macOS: erfordert mindestens 10.5 auf PowerPC oder 10.6 auf Intel Macs
- Raspberry Pi: erfordert mindestens einen Raspberry Pi 2 und Raspbian Jessie
- AmigaOS 4: erfordert SDL 2.0.12 (Juli 2020 oder neuer)
- MorphOS: erfordert mindestens MorphOS 3.8

1.4 Installation

Die Installation von RebelSDL ist unkompliziert und einfach: Kopieren Sie einfach die Datei `rebelsdl.hwp` für die Plattform in Hollywoods Plugin-Verzeichnis. Auf allen Systemen außer auf AmigaOS und kompatiblen müssen Plugins in einem Verzeichnis mit dem Namen `Plugins` gespeichert werden, das sich im selben Verzeichnis wie das Hauptprogramm von Hollywood befindet. Auf AmigaOS und kompatiblen Systemen müssen Plugins stattdessen in `LIBS:Hollywood` installiert werden. Unter macOS muss sich das Verzeichnis `Plugins` im Verzeichnis `Resources` des Programmpakets befinden, d.h. im Verzeichnis `HollywoodInterpreter.app/Contents/Resources`. Beachten Sie, dass `HollywoodInterpreter.app` im Programmpaket `Hollywood.app` selbst gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`.

Anschließend kopieren Sie den Inhalt des Ordners `Examples` in den Ordner `Examples` Ihrer Hollywood-Installation. Alle RebelSDL-Beispiele werden dann in Hollywoods GUI angezeigt und Sie können sie bequem über die Hollywood-GUI oder die IDE starten und anzeigen.

Unter Windows sollten Sie auch die Datei `RebelSDL.chm` in das Verzeichnis `Docs` Ihrer Hollywood-Installation kopieren. Dann können Sie die Onlinehilfe aufrufen, indem Sie F1 drücken, wenn sich der Cursor in der Hollywood IDE über einer RebelSDL-Funktion befindet.

Kopieren Sie unter Linux und macOS das Verzeichnis `RebelSDL`, das sich im Verzeichnis `Docs` des RebelSDL-Verteilungsarchivs befindet, in das Verzeichnis `Docs` Ihrer Hollywood-Installation. Beachten Sie, dass sich das Verzeichnis `Docs` unter macOS im Programmpaket `Hollywood.app` befindet, d.h. in `Hollywood.app/Contents/Resources/Docs`.

2 Über RebelSDL

2.1 Danksagungen

RebelSDL wurde von Andreas Falkenhahn geschrieben. Die Arbeit an diesem Projekt wurde im Sommer 2014 als Demonstration des leistungsfähigen neuen Display-Adapters und der Audio-Adapter-API von Hollywood 6.0 gestartet. Plugins können Hollywoods gesamten Display- und Audio-Handler übernehmen und durch einen eigenen Treiber ersetzen. Das Schreiben von RebelSDL half auch, Hollywoods Plugin-Schnittstelle zu konzipieren, um sie für neue Backends so flexibel wie möglich zu machen. Daher ist RebelSDL auch ein Testfall für die meisten Funktionen des Hollywood SDK. RebelSDL hat viele Funktionen, die optional aktiviert werden können, um bestimmte Funktionen des Hollywood SDK zu testen. Siehe Abschnitt 3.1 [RebelSDL aktivieren], Seite 9, für Details. Später wurde RebelSDL erweitert, um Zugriff auf andere SDL-Funktionen wie Joystick- und Gamecontroller-Unterstützung zu ermöglichen. Schließlich wurden einige Wrapper-Funktionen hinzugefügt, damit Skripte als SDL-Texturen auf Hollywood-Hardwarepinsel zugreifen können.

Ein besonderer Dank geht an Helmut Haake und Dominic Widmer 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, können Sie entweder eine E-Mail senden an `andreas@airsoftsoftwair.de` oder benutzen Sie das Kontaktformular auf `https://www.hollywood-mal.de/`.

2.2 Häufig gestellte Fragen

In diesem Abschnitt werden einige häufig gestellte Fragen behandelt. Bitte lesen Sie sie zuerst, bevor Sie in der Mailingliste oder im Forum nachfragen, da Ihr Problem hier möglicherweise behandelt wurde.

F: Warum ist RebelSDL so langsam auf meinem Raspberry Pi?

A: Beachten Sie, dass wenn Sie noch auf einem Raspberry Pi 1 arbeiten, nichts tun können, um Hardware-Beschleunigung von RebelSDL zu erhalten. Der Raspberry Pi 1 wird derzeit von RebelSDL nicht unterstützt. Sie brauchen mindestens ein Raspberry Pi 2. Stellen Sie zuerst sicher, dass Sie eine 2017 oder bessere Version von Raspbian Jessie oder Stretch verwenden. Stellen Sie dann sicher, dass Sie den experimentellen vc4 KMS/DRM OpenGL Treiber für X11 in `raspi-config` aktivieren. Siehe Abschnitt 3.10 [Raspberry Pi Besonderheiten], Seite 16, für Details.

F: Standardmäßig verwendet RebelSDL Direct3D unter Windows. Wie kann ich erzwingen, stattdessen OpenGL zu verwenden?

A: Setzen Sie den `RenderDriver`-Tag einfach auf `opengl`, wenn Sie RebelSDL mit `@REQUIRE` aktivieren. Siehe Abschnitt 3.1 [RebelSDL aktivieren], Seite 9, für Details.

F: Warum meldet RebelSDL Nicht-ASCII-Tasten nicht über OnKeyDown und OnKeyUp?

A: Das ist eine Einschränkung von SDL. Es unterstützt nur ASCII- und Steuertasten über die Ereignishandler `OnKeyDown` und `OnKeyUp`. Wenn Sie Hollywood 7.0 oder höher haben, können Sie einfach den Ereignishandler `VanillaKey` überwachen, um die echten Tastaturereignisse mit voller Unicode-Unterstützung zu erhalten.

F: Wie beende ich Skripte, die im Vollbildmodus laufen?

A: Drücken Sie einfach STRG+C. Dies funktioniert immer, außer wenn STRG+C explizit mit dem Hollywood-Befehl `CtrlCQuit()` deaktiviert wurde.

F: Gibt es ein Hollywood-Forum, in dem ich mich mit anderen Nutzern in Verbindung setzen kann?

A: Ja, bitte sehen Sie sich den Abschnitt "Forum" im offiziellen Hollywood Portal https://www.hollywood-mal.de/ unter "Hilfe" online an.

F: Wo kann ich um Hilfe bitten?

A: Es gibt ein lebhaftes englischsprachiges Forum auf http://forums.hollywood-mal.com und wir haben auch eine Mailing-Liste (auch englischsprachig), auf die Sie unter airsoft_hollywood@yahoogroups.com zugreifen können. Besuchen Sie https://www.hollywood-mal.de/ für Informationen darüber, wie Sie der Mailingliste beitreten können. Außerdem ist ein deutschsprachiges Forum vorhanden, welches Sie unter https://www.amiga-resistance.info/ erreichen können.

F: Ich habe einen Fehler gefunden.

A: Bitte posten Sie darüber in den speziellen Bereichen der Foren oder der Mailingliste.

2.3 Bekannte Probleme

Hier ist eine Liste von Dingen, die RebelSDL noch nicht unterstützt oder die in irgendeiner Weise verwirrend sein können:

- Die MorphOS-Version ist sehr instabil, z.B. der Vollbildmodus stürzt ab, das Iconieren eines Fensters stürzt ab, der Mauszeiger ist immer ausgeblendet usw.; Das ist nicht der Fehler von RebelSDL, sondern der SDL2-Port von MorphOS, der im Alpha-Zustand zu sein scheint. Ich habe versucht, mit dem MorphOS-Betreuer von SDL2 in Kontakt zu treten, habe aber keine Antwort erhalten, so dass Sie vorerst mit diesen Problemen leben müssen.
- Menüs werden nicht unterstützt
- Die Tastatur-Überwachung, die die Hollywood-Ereignisroutinen `OnKeyDown` und `OnKeyUp` zugeordnet ist, unterstützt derzeit nur ASCII- und Steuertasten. Dies liegt an einer Einschränkung in SDL, die kein feinabgestimmtes Überwachen unterstützt (d.h. Tasten drücken, Tasten wiederholen, Tasten loslassen) unter Verwendung internationaler Tastaturen; Benutzer, die Hollywood 7.0 oder besser haben, können stattdessen einfach den Ereignishandler `VanillaKey` verwenden. Dieser Ereignishandler wird echte Tastaturereignisse einschließlich vollständiger Unicode-Unterstützung bereitstellen.

– Nicht alle Displaystile werden unterstützt

2.4 Zukunft

Hier sind einige Dinge, die auf meiner Liste stehen:

– Unterstützung für weitere SDL-Aufrufe hinzufügen

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

2.5 Geschichte

In der Datei `history.txt` finden Sie ein vollständiges Änderungsprotokoll von RebelSDL, welches nur auf Englisch verfügbar ist.

3 Verwendung

3.1 RebelSDL aktivieren

Alles, was Sie tun müssen, damit Ihr Skript SDL anstelle des integrierten Grafiktreibers von Hollywood verwendet, ist die folgende Zeile am Anfang des Skripts hinzuzufügen:

```
@REQUIRE "rebelsdl"
```

Wenn Sie Hollywood von einer Konsole aus verwenden, können Sie Ihr Skript auch folgendermaßen starten:

```
Hollywood test.hws -requireplugins rebelsdl
```

Sobald das RebelSDL-Plugin für Ihr Skript aktiviert wurde, wird die gesamte Hollywood-Grafikausgabe über SDL umgeleitet. Beachten Sie, dass dies bei Skripte, die nicht für RebelSDL optimiert sind, normalerweise langsamer ist als der integrierte Grafiktreiber von Hollywood. Um eine optimale Leistung mit SDL zu erzielen, muss Ihr Skript einen hardwarebeschleunigten Doppelpuffer verwenden. Siehe Abschnitt 3.2 [Verwendung eines Hardware-Doppelpuffers], Seite 11, für Details.

RebelSDL akzeptiert die folgenden Argumente mit dem Aufruf `@REQUIRE`:

`EnableVSync:`
: Standardmäßig wird der Hardware-Doppelpuffer von RebelSDL mit der vertikalen Aktualisierung des Monitors synchronisiert. Dies bedeutet, dass `Flip()` immer bis zur nächsten vertikalen Aktualisierung blockiert und dann die Puffer umkehrt. Dies erzeugt perfekt glatte Grafiken, aber es bedeutet natürlich auch, dass Sie nicht schneller als die vertikale Aktualisierung des Monitors zeichnen können (normalerweise etwa 60 Mal pro Sekunde). Wenn Sie die vertikale Aktualisierung des Monitors ignorieren möchten, setzen Sie diesen Tag auf `False` und RebelSDL drosselt nicht den Wechsel des Doppel-Puffers. Er wird dann so schnell ausgeführt, wie es die Hardware ermöglicht. Der Standardwert ist `True`.

`ForceFullRefresh:`
: Wenn dieser Tag auf `False` gesetzt ist, aktualisiert RebelSDL nur die Teile des Displays, die tatsächlich geändert wurden. Dies ist zwar schneller, kann jedoch zu einigen Aktualisierungsproblemen führen, abhängig davon, wie das Skript die Grafiken zeichnet. Aus diesem Grund ist dieser Tag standardmäßig auf `True` gesetzt, was bedeutet, dass RebelSDL immer das gesamte Display aktualisiert, wenn etwas gezeichnet wird. Dies ist zwar langsamer, garantiert aber, dass es keine visuelle Artefakte gibt, da die vorderen und hinteren Puffer immer vollständig synchron sind.

`RenderDriver:`
: Mit diesem Tag können Sie einen anderen Wiedergabe-Treiber als den Standard-Wiedergbabe-Treiber auswählen. Dies ist vor allem für Testzwecke nützlich. Beispielsweise ist es mit diesem Tag möglich, RebelSDL zu zwingen, OpenGL unter Windows anstelle des Standard-Direct3D-Treibers zu verwenden. Mögliche Werte für diesen Tag sind `direct3d`, `opengl`, `opengles`, `opengles2`, `rpi` und `software`. Siehe Abschnitt 3.10 [Raspberry Pi Besonderheiten], Seite 16, für weitere Informationen über den RPI-Treiber.

`UseAudioAdapter:`
: Standardmäßig wird RebelSDL den integrierten Audiotreiber von Hollywood durch einen benutzerdefinierten Audiotreiber ersetzen, der SDL zum Abspielen von Audio verwendet. Wenn Sie das nicht möchten, setzen Sie diesen Tag auf `False`. Dann wird Hollywoods eingebauter Audiotreiber verwendet, selbst wenn RebelSDL aktiv ist. Normalerweise ist es jedoch nicht erforderlich, diesen Tag festzulegen, es sei denn, Sie haben Probleme mit dem Audiotreiber von RebelSDL. Der Standardwert ist `True`.

`UseBitmapAdapter:`
: Wenn dies auf `True` gesetzt ist, wird RebelSDL Hollywoods integrierten Handler für Software-Bitmaps überschreiben. Dies hat keine praktischen Vorteile und wurde nur implementiert, um die entsprechende Hollywood SDK-Funktionalität zu testen. Standardeinstellung ist `False`.

`UseDesktopFullScreen:`
: SDL bietet einen speziellen Anzeigemodus, bei dem von SDL geöffnete Fenster automatisch auf die Abmessungen des Desktops skaliert werden. Das Fenster belegt dann den gesamten Bildschirmbereich, ohne die Auflösung des Monitors zu ändern. Sie können diesen Modus aktivieren, indem Sie diesen Tag auf `True` setzen. Somit wird die automatische Skalierung für Ihren Bildschirm aktiviert. Beachten Sie, dass ein ähnlicher Effekt erzielt werden kann, wenn Sie den `FullScreenScale-Anzeigemodus` von Hollywood verwenden. Es ist jedoch vorzuziehen, `UseDesktopFullScreen` zu verwenden, da dieser direkt mit SDL verknüpft ist. Standardeinstellung ist `False`.

`UseDoubleBufferAdapter:`
: Wenn dies auf `False` gesetzt ist, unterstützt RebelSDL keine Hardware-Doppelpuffer. Da Hardware-Doppelpuffer eine der wichtigsten Funktionen von RebelSDL sind, gibt es wahrscheinlich keinen Fall, in dem Sie diese Funktion deaktivieren möchten. Es kann aber für Debugging-Zwecke verwendet werden. Der Standardwert ist `True`.

`UseSoftwareRenderer:`
: Standardmäßig versucht SDL, die GPU zu verwenden, um Grafiken wann und wo immer möglich zu zeichnen. Wenn Sie dies nicht möchten, können Sie diesen Tag auf `True` setzen, um SDL in den reinen Software-Zeichenmodus zu versetzen. Dies ist wahrscheinlich nur zum Testen und Debuggen von Nutzen, da normalerweise die Hardware-Wiedergabe für die beste Leistung verwendet werden sollte. Standardeinstellung ist `False`.

`UseVideoBitmapAdapter:`
: Wenn dies auf `False` gesetzt ist, unterstützt RebelSDL keine Hardwarepinsel. Da Hardwarepinsel eine der wichtigsten Funktionen von RebelSDL sind, gibt es wahrscheinlich keinen Fall, in dem Sie diese Funktion deaktivieren möchten. Es kann aber für Debugging-Zwecke verwendet werden. Der Standardwert ist `True`.

Hier ist ein Beispiel, wie Argumente an den Präprozessorbefehl `@REQUIRE` übergeben werden:

```
@REQUIRE "rebelsdl", {UseDesktopFullScreen = True}
```

Alternativ können Sie auch das Konsolen Argument `-requiretags` verwenden, um diese Argumente zu übergeben. Weitere Informationen finden Sie im Hollywood-Handbuch.

3.2 Verwendung eines Hardware-Doppelpuffers

Wenn Sie möchten, dass Ihr Skript von den hardwarebeschleunigten Zeichenfunktionen von RebelSDL profitiert, müssen Sie einen Hardware-Doppelpuffer verwenden und alle Ihre Zeichnungen innerhalb dieses Doppelpuffers ausführen. Durch die Verwendung eines Hardware-Doppelpuffers wird außerdem sichergestellt, dass die Grafikausgabe mit der Aktualisierungsrate des Monitors synchronisiert wird, um Flimmern zu vermeiden. Um eine optimale Leistung mit RebelSDL zu erhalten, sollte Ihre Hauptschleife immer so aussehen:

```
@REQUIRE "rebelsdl"

BeginDoubleBuffer(True)  ; Richtet einen Hardware-Doppelpuffer ein

Repeat
    ....           ; zeichnet hier das nächste Einzelbild
    Flip()         ; wartet auf die vertikale Aktualisierung,
                   ; und tauscht dann die Puffer aus
    CheckEvent()   ; Ereignis-Callback ausführen
Forever
```

Der Aufruf von `CheckEvent()` ist nur erforderlich, wenn Ihr Skript auf Ereignishandler warten muss, die mit `InstallEventHandler()` installiert wurden. Beachten Sie, dass Sie das nächste Einzelbild nicht in einem Intervall-Callback zeichnen sollten, welcher mit einer konstanten Bildrate (etwa 50 fps) läuft. Dies garantiert nicht, dass die Zeichnung mit der vertikalen Aktualisierung synchronisiert wird, da verschiedene Monitore unterschiedliche Bildwiederholraten verwenden und somit erhalten Sie flackernde Grafiken. Wenn Sie Ihre Zeichnung wie oben beschrieben ausführen, können Sie sicher sein, dass die vorderen und hinteren Puffer in perfekter Synchronisation mit der vertikalen Aktualisierung des Monitors getauscht werden.

Darüber hinaus müssen Sie darauf achten, wie Sie Ihre Grafiken tatsächlich zeichnen, da die meisten Zeichnungsbefehle von Hollywood vollständig im Softwaremodus arbeiten und daher nicht von der Hardwarebeschleunigung profitieren. Siehe Abschnitt 3.3 [Zeichnen von Grafiken], Seite 11, für Details.

Stellen Sie beim Zeichnen von Pinseln in einem Hardware-Doppelbuffer sicher, dass Sie nur Hardwarepinsel verwenden, da nur diese direkt mithilfe der Hardwarebeschleunigung gezeichnet werden können. Es ist auch möglich, normale Pinsel auf Hardware-Doppelpuffer zu zeichnen, aber das wird sehr langsam. Siehe Abschnitt 3.4 [Verwendung von Hardwarepinseln], Seite 12, für Details.

Wichtig: SDL ist für die Verwendung des Doppelpuffers vorgesehen. Daher können Sie nur von der Hardwarebeschleunigung profitieren, wenn Sie in einem Doppelpuffer zeichnen. Das Zeichnen außerhalb eines Doppelpuffers ist möglich, aber es wird viel langsamer.

3.3 Zeichnen von Grafiken

Um eine optimale Leistung zu erzielen, müssen Sie beim Zeichnen Ihrer Grafiken sehr vorsichtig sein. Die meisten von Hollywoods Zeichenbefehlen sind nur in Software implemen-

tiert, d.h. Sie zeichnen mit der CPU statt mit der GPU. Dies kann insbesondere bei langsameren CPUs zu einem Engpass werden. Daher sollten Sie wissen, welche Zeichenfunktionen hardwarebeschleunigt sind und welche nicht und schreiben Ihre Skripte entsprechend.

Die folgenden Hollywood-Befehle werden hardwarebeschleunigt, wenn RebelSDL aktiv ist und sie in einem Hardware-Doppelpuffer verwendet werden:

```
Box()
Cls()
Line()
Plot()
DisplayBrush()
```

`DisplayBrush()` wird die Hardwarebeschleunigung nur verwenden, wenn er mit einem Hardwarepinsel aufgerufen wird. Siehe Abschnitt 3.4 [Verwendung von Hardwarepinseln], Seite 12, für Details. Wenn `DisplayBrush()` mit einem Software-Pinsel verwendet wird, d.h. einem Pinsel, der sich nicht im Videospeicher befindet, zeichnet er den Pinsel mit der CPU, die viel langsamer ist.

`Box()`, `Line()` und `WritePixel()` werden nur Hardwarebeschleunigung verwenden, wenn der Füllstil entweder `#FILLNONE` oder `#FILLCOLOR` ist und keine anderen Formstile wie `#EDGE` oder `#SHADOW` aktiv sind. Sobald Sie mit anderen Füll- oder Formstilen zeichnen möchten, greifen diese Befehle auf ihr Software-Pendant zurück und sind daher sehr langsam. Sie können dieses Problem umgehen, indem Sie zuerst die Grafiken in einem Hardwarepinsel gestalten und dann nur diesen Hardwarepinsel zeichnen. Dies ist eine gute Strategie, da Hollywoods Software-Wiedergabe nur einmal verwendet wird, d.h. wenn die Grafiken in einen Hardwarepinsel gezeichnet werden. Danach profitieren Sie immer von der Hardwarebeschleunigung, da Hardwarepinsel wirklich schnell gezeichnet werden können.

Vergessen Sie schließlich nicht, dass Sie alle Ihre Zeichnung in einer Hardware-Doppelpufferschleife verwenden sollten. Siehe Abschnitt 3.2 [Verwendung eines Hardware-Doppelpuffers], Seite 11, für Details.

3.4 Verwendung von Hardwarepinseln

RebelSDL unterstützt die Erstellung von Hardwarepinseln. Hardwarepinsel befinden sich im GPU-Speicher und können somit in kürzester Zeit gezeichnet werden. Auf den meisten Grafikkarten können sie auch sehr effizient von der GPU skaliert und transformiert werden. Damit Hollywood einen Hardwarepinsel erstellt, müssen Sie lediglich den optionalen `Hardware-Tag` auf `True` setzen. Dieser Tag wird von den meisten Hollywood-Befehlen unterstützt, die Pinsel erstellen.

Hier ist ein Beispiel:

```
@REQUIRE "rebelsdl"  ; Muss die erste Zeile sein
@BRUSH 1, "sprites.png", {Hardware = True}
```

Im obigen Code erstellt RebelSDL den Pinsel 1 im Videospeicher. Er kann dann fast ohne Aufwand mit der GPU gezeichnet werden. Beachten Sie jedoch, dass Hardwarepinsel nur auf Hardware-Doppelpuffer gezeichnet werden können. Siehe Abschnitt 3.2 [Verwendung eines Hardware-Doppelpuffers], Seite 11, für Details.

Um einen Hardwarepinsel zu transformieren, können Sie die Befehle `ScaleBrush()`, `RotateBrush()` und `TransformBrush()` verwenden. Transformationen von Hardwarepin-

seln sind in der Regel auch GPU-beschleunigt und damit um ein Vielfaches schneller als die von der CPU durchgeführten Transformationen.

Beachten Sie, dass Hardwarepinsel nur auf dem Display gezeichnet werden kann, welches bei der Zuweisung angegeben wurde. Wenn Ihr Skript also mehrere Displays verwendet, müssen Sie Hollywood den Identifikator des Displays mitteilen, mit dem Sie diesen Hardwarepinsel verwenden möchten. Dies kann durch Angabe des Tags "Display" nach dem Tag "Hardware" erfolgen. Hier ist ein Beispiel:

```
@REQUIRE "rebelsdl"  ; Muss die erste Zeile sein
@DISPLAY 1, {Title = "First display"}
@DISPLAY 2, {Title = "Second display"}
@BRUSH 1, "sprites.png", {Hardware = True, Display = 1}
@BRUSH 2, "sprites.png", {Hardware = True, Display = 2}
```

Der obige Code wird den Pinsel 1 so zuweisen, dass er auf dem Display 1 gezeichnet werden kann und er wird den Pinsel 2 so zuweisen, dass er auf dem Display 2 gezeichnet werden kann. Es ist jedoch nicht möglich, den Pinsel 2 auf dem Display 1 zu zeichnen oder Pinsel 1 auf dem Display 2 anzuzeigen! RebelSDL-Hardwarepinsel sind immer Displayabhängig und können nur auf dem Display gezeichnet werden, für den sie zugewiesen wurden.

Weitere Informationen zu Hardwarepinseln und Hardware-Doppelpuffern finden Sie im Hollywood-Handbuch.

Sie können RebelSDL auch als Hilfs-Plugin verwenden, um unter Windows, macOS und Linux Hardwarepinsel-Unterstützung für Hollywood hinzuzufügen. Standardmäßig unterstützt Hollywood keine Hardwarepinsel auf diesen Systemen, aber RebelSDL kann diese Funktion zu Hollywood hinzufügen. Siehe Abschnitt 3.9 [RebelSDL als Hilfs-Plugin], Seite 15, für Details.

Das Beispielskript `SmoothScroll.hws`, das mit RebelSDL ausgeliefert wird, demonstriert die Verwendung von Hardwarepinseln und einem Hardware-Doppelpuffer, um ein butterweiches Scrollen zu erreichen, welches vollständig mit der vertikalen Aktualisierung des Monitors synchronisiert ist.

3.5 Offscreen-Wiedergabe

Wenn Sie `True` an den Befehl `sdl.EnableOffscreenRender()` übergeben, können alle nach diesem Befehl erstellten Hardwarepinsel mit dem Befehl `SelectBrush()` von Hollywood gezeichnet werden. Diese Zeichenoperationen können auch hardwarebeschleunigt werden, was ihnen einen Vorteil gegenüber den standardmäßigen Zeichenroutinen von Hollywood gibt, die nur auf Softwarepinsel zurückgreifen können.

Es gibt jedoch eine wichtige Einschränkung, die Sie beachten sollten: Hardwarepinsel, die mit `SelectBrush()` gezeichnet werden können, verlieren ihren Inhalt unter Windows, wenn die Fenstergröße oder der Anzeigemodus des Fensters geändert wird, z.B. vom Vollbild- zum Fenstermodus oder umgekehrt. Die Verwendung von Hardwarepinseln, die wirklich gezeichnet werden können, ist daher nur dann sinnvoll, wenn Sie sie bei jedem Bild aktualisieren und bei jeder Änderung der Fenstergröße oder des Anzeigemodus neu initialisieren können. Wenn Ihr Skript das nicht kann, werden Sie in Schwierigkeiten geraten. Wegen dieser Einschränkung ist es auch dringend ratsam, das Argument bei `sdl.EnableOffscreenRender()` auf `False` zu setzen, unmittelbar nachdem Sie die Hardwarepinsel erstellt haben, die Sie für das Offscreen-Zeichnen verwenden möchten. Andernfalls werden alle Hardwarepinsel, die

anschließend erstellt werden, auch für das Offscreen-Zeichnen vorbereitet und unterliegen daher den oben beschriebenen Einschränkungen.

Hinweis: Um die Hardwarebeschleunigung beim Offscreen-Zeichnen zu verwenden, müssen Sie die gleichen Regeln befolgen wie beim Zeichnen in einen Hardware-Doppelpuffer. Nur wenige Grafikoperationen werden hardwarebeschleunigt. Siehe Abschnitt 3.3 [Zeichnen von Grafiken], Seite 11, für Details.

Es ist auch wichtig zu beachten, dass die verschiedenen von `SelectBrush()` unterstützten Kombinationsmodi nicht mit RebelSDL funktionieren. Stattdessen werden Grafiken immer auf die Farbe und den Alphakanal des Hardwarepinsels gezeichnet, unabhängig davon, welchen Modus Sie angeben. Außerdem können Sie `SelectMask()` oder `SelectAlphaChannel()` nicht mit Hardwarepinseln verwenden. Es ist nur möglich, mit `SelectBrush()` gleichzeitig in Farb- und Alphakanäle zu zeichnen.

3.6 Verwendung der SDL-Wiedergabe

Fortgeschrittene Benutzer können mit RebelSDL die SDL-Wiedergabe-Funktionen direkt aufrufen, um ultimative Flexibilität zu erhalten. Beachten Sie jedoch, dass Sie bei dieser Vorgehensweise auf einem sehr niedrigen Niveau arbeiten. Einige Hollywood-Funktionen wie das Autoskalierungs-System funktionieren nicht mehr, da alle Ihre Zeichnungsaufrufe direkt über SDL geleitet werden, ohne dass Hollywood eine Chance hat einzugreifen. Dies führt zu sehr wenig Zusatzarbeit (Overhead), geht aber auf Kosten bestimmter Hollywood-Funktionen wie dem Autoskalierungs-System, die nicht mehr funktionieren.

Wenn Sie SDL-Wiedergabe-Aufrufe direkt ausführen, sollten Sie sie auch nicht mit Hollywood-Zeichnungsaufrufen mischen. Sie sollten entweder vollständig mit Hollywood-Funktionen oder mit SDL-Wiedergabe-Funktionen zeichnen. Das Mischen von beiden ist möglich, kann aber zu unerwarteten Ergebnissen führen, da Hollywood-Zeichnungsfunktionen natürlich die Statusinformationen der SDL-Wiedergabe selbst ändern. D.h. wenn Sie mit dem Aufruf von `sdl.SetRenderDrawColor()` die Zeichnungsfarbe auf rot setzen und dann mit dem Hollywood-Befehl `Box()` eine blaue Box zeichnen, dann wird die Wiedergabefarbe plötzlich blau, weil der Aufruf von `Box()` die Wiedergabefarbe in blau geändert hat. Dies sind Nebenwirkungen, die Sie beim Mischen von SDL-Wiedergabe-Aufrufen und Hollywood-Zeichnungsaufrufen bewältigen müssen.

3.7 Joysticks und Gamecontroller

Mit RebelSDL können Sie auf die umfassenden Joystick- und Gamecontroller-Funktionen von SDL zugreifen. Für die beste Kompatibilität wird empfohlen, eine Datei mit dem Namen `gamecontrollerdb.txt` im selben Verzeichnis wie Ihr Skript zu speichern. Diese Datei muss Kalibrierungsinformationen für die verschiedenen Joysticks und Gamecontroller enthalten. Sie finden eine von der Gemeinschaft gepflegte Version dieser Datei hier: `https://github.com/gabomdq/SDL_GameControllerDB`

3.8 Erhöhung der Ausführungsgeschwindigkeit

Um die Geschwindigkeit der rohen Ausführung Ihres Skripts zu erhöhen, können Sie die interne Funktion "LineHook" von Hollywood mit dem Befehl `DisableLineHook()` deaktivieren und mit `EnableLineHook()` wieder aktivieren. Dies wird die

Ausführungsgeschwindigkeit Ihres Skripts erheblich verbessern, falls beim Zeichnen des nächsten Einzelbildes viel Hollywood-Code ausgeführt werden muss. Beachten Sie jedoch, dass Sie "LineHook" für jedes von Ihnen gezeichnete Einzelbild abermals aktivieren müssen oder Ihr Fenster reagiert nicht mehr. So könnte eine geschwindigkeitsoptimierte Implementierung der Hauptschleife aussehen:

```
@REQUIRE "rebelsdl"

BeginDoubleBuffer(True)  ; Richtet einen Hardware-Doppelpuffer ein

Repeat
    DisableLineHook() ; deaktivieren Sie den LineHook, während Sie den
                      ; nächsten Rahmen zeichnen
    p_DrawFrame()     ; zeichnet hier den nächsten Rahmen
    EnableLineHook()  ; aktiviert den LineHook erneut
    Flip()            ; wartet auf die vertikale Aktualisierung,
                      ; und tauscht dann die Puffer aus
    CheckEvent()      ; Ereignis-Callback ausführen
Forever
```

Beachten Sie, dass Sie hier nur einen Geschwindigkeitsunterschied bemerken, wenn `p_DrawFrame()` viele Zeilen Hollywood-Code ausführt. Wenn `p_DrawFrame()` nur aus 20 Codezeilen besteht, werden Sie keinen Unterschied bemerken. Es ist nur mit Hunderten von Codezeilen oder langen Schleifen bemerkbar.

Weitere Informationen finden Sie in der Dokumentation von `DisableLineHook()` und `EnableLineHook()` im Hollywood-Handbuch.

3.9 RebelSDL als Hilfs-Plugin

RebelSDL kann auch als Hilfs-Plugin verwendet werden, um das Problem zu umgehen, dass Hollywood nur auf AmigaOS und Kompatible hardwarebeschleunigte Doppelpuffer und Pinsel unterstützt und unter Windows, macOS oder Linux nicht. Wenn Sie jedoch `@REQUIRE` "rebelsdl" installieren, wird Hardware-Doppelpuffer und Hardwarepinsel auch unter Windows, macOS und Linux zur Verfügung stehen, da RebelSDL dies unterstützt.

Daher können Sie RebelSDL auch als Hilfs-Plugin verwenden, um hardwarebeschleunigte Doppelpuffer-Unterstützung für Windows, macOS und Linux zu erhalten. Sie müssen nicht einen der SDL-Befehle direkt verwenden. Sie können einfach `@REQUIRE` "rebelsdl" aufrufen, einen Hardware-Doppelpuffer einrichten und dann mit Hardwarepinseln zeichnen. Auf diese Weise können Sie die Hardwarebeschleunigung nutzen, ohne eine einzige Zeile SDL-Code schreiben zu müssen!

Auf AmigaOS und kompatiblen Geräten ist dies nicht notwendig, da Hollywood hardwarebeschleunigte Doppelpuffer und Pinsel standardmäßig bereits unterstützt. Dennoch kann die Verwendung von RebelSDL auf AmigaOS als Hardware-Doppelpuffer-Treiber im Vollbildmodus nützlich sein, da RebelSDL das Zeichnen perfekt mit der vertikalen Aktualisierung des Monitors synchronisiert, so dass es normalerweise besser aussieht als die von Hollywood direkt verwalteten Doppelpuffer.

Siehe Abschnitt 3.2 [Verwenden eines Hardware-Doppelpuffers], Seite 11, für Details. Siehe Abschnitt 3.4 [Verwendung von Hardwarepinseln], Seite 12, für Details.

Das Skript `SmoothScroll.hws`, das mit RebelSDL ausgeliefert wird, demonstriert die Verwendung von Hardwarepinseln und einem Hardware-Doppelpuffer, um ein butterweiches Scrollen zu erreichen, das vollständig mit der vertikalen Aktualisierung des Monitors synchronisiert ist.

3.10 Raspberry Pi Besonderheiten

RebelSDL kann auf dem Raspberry Pi wegen seiner vergleichsweise schlechten CPU sehr nützlich sein, denn die in Hollywood integrierte CPU-Wiedergabe ist darauf sehr langsam. Mit RebelSDL erhalten Sie hardwarebeschleunigtes Zeichnen und Skalieren, was die Leistung Ihres Skripts erheblich steigern kann. Es gibt jedoch einige Dinge, die Sie beachten müssen. Für den Raspberry Pi stehen zunächst zwei völlig unterschiedliche Treiber zur Verfügung:

1. VideoCore 4 OpenGL driver (KMS/DRM): Seit 2017 wird Raspbian Jessie mit einem experimentellen vc4 KMS/DRM OpenGL Treiber ausgeliefert. Um diesen Treiber zu aktivieren, müssen Sie den `sudo raspi-config` ausführen, indem Sie zu den `Advanced Options` Einstellungen gehen und dann `GL (Full KMS)` und anschließend `GL Driver` im Menü auswählen. Sobald Sie Ihr System neu gestartet haben, wird X11 dann diesen neuen vc4-Treiber verwenden und RebelSDL wird ihn auch benutzen können. Sie müssen aber sicherstellen, dass Sie eine aktuelle Version von Raspbian Jessie (2017 oder neuer) oder Raspbian Stretch verwenden. Ältere Versionen von Jessie haben diesen Treiber noch nicht. Beachten Sie auch, dass die Aktivierung des neuen vc4-Treibers ab September 2017 die HDMI-Audioausgabe von Jessie unterbricht (nicht bei Stretch). Ab September 2017 können Sie nur dann Audio über den Kopfhörerausgang von Jessie erhalten, wenn der vc4-Treiber aktiviert ist.
2. Raspberry Pi native driver: Dies ist ein alternativer Treiber, der direkt auf die Grafikhardware des Pi unter vollständiger Umgehung von OpenGL und X11 zugreift. Dies bedeutet, dass Sie Ihre Skripte auch außerhalb von X11 ausführen können. Dieser Treiber funktioniert auch mit älteren Versionen von Raspbian Jessie, aber nur wenn der experimentelle OpenGL-Treiber (siehe oben) deaktiviert ist. Um den nativen Raspberry-Pi-Wiedergabe zu aktivieren, müssen Sie `rpi` an den `RenderDriver`-Tag übergeben, wenn Sie `@REQUIRE` "rebelsdl" aufrufen:

   ```
   @REQUIRE "rebelsdl", {RenderDriver = "rpi"}
   ```

   RebelSDL verwendet dann die native Raspberry Pi-Wiedergabe. Beachten Sie, dass die native Pi-Wiedergabe außerhalb von X11 immer den gesamten Bildschirm übernimmt. Es ist nicht möglich, die native Pi-Wiedergabe im Fenstermodus auszuführen. Es wird immer der gesamte Bildschirm ausgefüllt. Alle Hollywood-Funktionen, die ein Fenster benötigen, funktionieren auch nicht mit der nativen Pi-Wiedergabe.

Standardmäßig verwendet RebelSDL den vc4 OpenGL-Treiber und wenn dieser nicht vorhanden ist, greift er auf den OpenGL-Modus zurück, der natürlich sehr langsam ist. Die native Pi-Wiedergabe wird nur aktiviert, wenn Sie ihn explizit wie oben beschrieben anfordern.

4 Beispiele

4.1 Beispiele

RebelSDL enthält eine Reihe von Beispielen, die zeigen, wie man das Plugin benutzt und es Ihnen erlauben sollte, wirklich schnell zu beginnen. Hier ist eine Liste von Beispielen, die mit RebelSDL verteilt werden:

Aladdin Ein RebelSDL-Port des Prodigy Cracktro für Aladdin.

BeastScroll
: Ein Remake des berühmten Shadow of the Beast Scrollers in RebelSDL.

CannonFodder
: Ein RebelSDL-Port des Cannon Fodder Cracktro von Crystal.

Creatures Ein RebelSDL-Port des Creatures Cracktro von Crystal.

Dynablaster
: Ein RebelSDL-Port des Dynablaster Cracktro von Vision Factory.

GPUScale Demonstriert, wie GPU-beschleunigte Skalierung und Rotation mit RebelSDL verwendet wird.

Lemmings Ein RebelSDL-Port des Lemmings Cracktro von Skid Row.

Moonstone
: Ein RebelSDL-Port des Moonstone-Cracktro von Crystal.

MultiDisplays
: Demonstriert, wie mehrere Displays mit RebelSDL verwendet werden.

Pang Ein RebelSDL-Port des Pang Cracktro von Horizon.

PPHammer
: Ein RebelSDL-Port des PP Hammer Cracktro von Crystal.

SmoothScroll
: Demonstriert, wie man mit Hardwarepinseln ein perfektes Scrollen erreicht.

SteelEmpire
: Ein RebelSDL-Port des Steel Empire Cracktro von Crystal.

Superfrog Ein RebelSDL-Port des Superfrog Cracktro von Crystal.

SuperStardust
: Ein RebelSDL-Port des Super Stardust Cracktro von Prestige.

Turrican2 Ein RebelSDL-Port des Turrican 2 Cracktro von Defjam.

Turrican3 Ein RebelSDL-Port des Turrican 3 Cracktro von Hoodlum.

Zool Ein RebelSDL-Port des Zool Cracktro von Crystal.

5 Joystick-Referenz

5.1 sdl.ForceJoystickMode

BEZEICHNUNG

sdl.ForceJoystickMode – erzwingt den Gamecontroller in den Joystick-Modus

ÜBERSICHT

```
sdl.ForceJoystickMode(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Gamecontroller am angegebenen Port in den Joystick-Modus zu versetzen. Sie können dann den Status des Gamecontrollers abfragen, als wäre es ein Joystick.

EINGABEN

`port` Game-Port, der verwendet werden soll

5.2 sdl.GetAxis

BEZEICHNUNG

sdl.GetAxis – fragt den Status der angegebenen Achse ab

ÜBERSICHT

```
state = sdl.GetAxis(port, axis)
```

BESCHREIBUNG

Wenn das Gerät am angegebenen Port ein Joystick ist, muss `axis` die Nummer der abzufragenden Achse sein. 0 wird typischerweise für die x-Achse und 1 für die y-Achse verwendet. Einige Joysticks verwenden die Achsen 2 oder 3 für zusätzliche Tasten. Sie können `sdl.GetNumAxes()` verwenden, um die Anzahl der Achsen zu ermitteln.

Wenn das Gerät am angegebenen Port ein Gamecontroller ist, muss `axis` eine der folgenden vordefinierten Konstanten sein:

```
#SDL_AXIS_LEFTX
#SDL_AXIS_LEFTY
#SDL_AXIS_RIGHTX
#SDL_AXIS_RIGHTY
#SDL_AXIS_TRIGGERLEFT
#SDL_AXIS_TRIGGERRIGHT
```

Der Rückgabewert ist ein Wert zwischen -32768 und 32767. Es kann erforderlich sein, diesen Werten bestimmte Toleranzen zuzuordnen, um Jitter (Schwankungen) zu berücksichtigen. Beachten Sie, dass Gamecontroller-Auslöser jedoch zwischen 0 und 32767 liegen. Sie geben niemals einen negativen Wert zurück.

EINGABEN

`port` Game-Port, der abgefragt werden soll

`axis` die abzufragende Achse

RÜCKGABEWERTE

`state` Status der angegebenen Achse (typischerweise -32768 bis 32767)

5.3 sdl.GetBall

BEZEICHNUNG

sdl.GetBall – fragt den Status der angegebenen Kugel ab

ÜBERSICHT

```
dx, dy = sdl.GetBall(port, ball)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Änderung der Kugelachse seit der letzten Abfrage zu erhalten. Dies ist nur für Joysticks möglich, nicht für Gamecontroller.

Sie müssen den Kugelindex übergeben, um ihn in `ball` abzufragen. Kugelindizes beginnen bei Index 0. `sdl.GetBall()` gibt die Differenz in der x- und y-Achsenposition seit der letzten Abfrage zurück. Beachten Sie, dass diese Rückgabewerte Deltawerte sind, da Trackballs nur relative Bewegungen zurückgeben können.

EINGABEN

`port` Game-Port, der abgefragt werden soll

`ball` Kugelindex, der abgefragt werden soll

RÜCKGABEWERTE

`dx` der Unterschied in der x-Achsenposition seit der letzten Abfrage

`dy` der Unterschied in der Y-Achsenposition seit der letzten Abfrage

5.4 sdl.GetButton

BEZEICHNUNG

sdl.GetButton – fragt den Status der angegebenen Taste ab

ÜBERSICHT

```
state = sdl.GetButton(port, button)
```

BESCHREIBUNG

Wenn das Gerät am angegebenen Port ein Joystick ist, muss `button` der Index der gewünschten Taste sein (beginnend mit 0). Die Anzahl der Joystick-Tasten kann durch Aufrufen von `sdl.GetNumButtons()` ermittelt werden.

Wenn das Gerät am angegebenen Port ein Gamecontroller ist, muss `button` eine der folgenden vordefinierten Konstanten sein:

```
#SDL_BUTTON_A
#SDL_BUTTON_B
#SDL_BUTTON_X
#SDL_BUTTON_Y
#SDL_BUTTON_BACK
```

```
#SDL_BUTTON_GUIDE
#SDL_BUTTON_START
#SDL_BUTTON_LEFTSTICK
#SDL_BUTTON_RIGHTSTICK
#SDL_BUTTON_LEFTSHOULDER
#SDL_BUTTON_RIGHTSHOULDER
#SDL_BUTTON_DPAD_UP
#SDL_BUTTON_DPAD_DOWN
#SDL_BUTTON_DPAD_LEFT
#SDL_BUTTON_DPAD_RIGHT
```

EINGABEN

`port` Game-Port, der abgefragt werden soll

`button` Taste, die abgefragt werden soll

RÜCKGABEWERTE

`state` Zustand der angegebenen Taste (1 für gedrückten Zustand, 0 nicht gedrückten Zustand)

5.5 sdl.GetHat

BEZEICHNUNG

sdl.GetHat – fragt den Status des angegebenen Rundblickschalters ab

ÜBERSICHT

```
state = sdl.GetHat(port, hat)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den aktuellen Status eines Rundblickschalters (Coolie Hat) auf einem Joystick zu erhalten. Dies ist nur für Joysticks möglich, nicht für Gamecontroller. Sie müssen den Rundblickschalter-Index übergeben, um den Status abzurufen. Rundblickschalter-Indizes beginnen bei Index 0.

Der Rückgabewert ist eine der folgenden vordefinierten Konstanten:

```
#SDL_HAT_CENTERED
#SDL_HAT_UP
#SDL_HAT_RIGHT
#SDL_HAT_DOWN
#SDL_HAT_LEFT
#SDL_HAT_RIGHTUP
#SDL_HAT_RIGHTDOWN
#SDL_HAT_LEFTUP
#SDL_HAT_LEFTDOWN
```

EINGABEN

`port` Game-Port, der abgefragt werden soll

`hat` Rundblickschalter, dessen Index sie erhalten wollen

RÜCKGABEWERTE

`state` Status des angegebenen Rundblickschalters

5.6 sdl.GetJoysticks

BEZEICHNUNG

sdl.GetJoysticks – ruft die Anzahl der verfügbaren Joysticks ab

ÜBERSICHT

```
n = sdl.GetJoysticks()
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Anzahl der an das System angeschlossenen Joysticks zu erhalten.

EINGABEN

keine

RÜCKGABEWERTE

`n` Anzahl der angehängten Joysticks

5.7 sdl.GetNumAxes

BEZEICHNUNG

sdl.GetNumAxes – gibt die Anzahl der Joystickachsen zurück

ÜBERSICHT

```
num = sdl.GetNumAxes(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Anzahl der allgemeinen Achsensteuerungen auf einem Joystick abzurufen. Dies ist nur für Joysticks möglich, nicht für Gamecontroller.

EINGABEN

`port` Game-Port, der abgefragt werden soll

RÜCKGABEWERTE

`num` Anzahl der Joystickachsen

5.8 sdl.GetNumBalls

BEZEICHNUNG

sdl.GetNumBalls – gibt die Anzahl der Joystick-Kugeln zurück

ÜBERSICHT

```
num = sdl.GetNumBalls(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Anzahl der Trackballs auf einem Joystick zu erhalten. Dies ist nur für Joysticks möglich, nicht für Gamecontroller.

EINGABEN

`port` Game-Port, der abgefragt werden soll

RÜCKGABEWERTE

`num` Anzahl der Joystick-Trackballs

5.9 sdl.GetNumButtons

BEZEICHNUNG

sdl.GetNumButtons – gibt die Anzahl der Joystick-Tasten zurück

ÜBERSICHT

```
num = sdl.GetNumButtons(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Anzahl der Tasten auf einem Joystick zu erhalten. Dies ist nur für Joysticks möglich, nicht für Gamecontroller.

EINGABEN

`port` Game-Port, der abgefragt werden soll

RÜCKGABEWERTE

`num` Anzahl der Joystick-Tasten

5.10 sdl.GetNumHats

BEZEICHNUNG

sdl.GetNumHats – gibt die Anzahl der Joystick-Rundblickschalter zurück

ÜBERSICHT

```
num = sdl.GetNumHats(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Anzahl der Rundblickschalter (Coolie Hat) auf einem Joystick zu erhalten. Dies ist nur für Joysticks möglich, nicht für Gamecontroller.

EINGABEN

`port` Game-Port, der abgefragt werden soll

RÜCKGABEWERTE

`num` Anzahl der Joystick-Rundblickschalter

5.11 sdl.IsGameController

BEZEICHNUNG

sdl.IsGameController – überprüft, ob das Eingabegerät ein Gamecontroller ist

ÜBERSICHT

```
res = sdl.IsGameController(port)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um herauszufinden, ob das Eingabegerät am angegebenen Port ein Gamecontroller oder ein Joystick ist. Der Befehl gibt `True` für einen Gamecontroller und `False` für einen Joystick zurück.

EINGABEN

`port`	Game-Port, der abgefragt werden soll

RÜCKGABEWERTE

`res`	boolescher Wert

6 Tastatur-Referenz

6.1 sdl.SetTextInputRect

BEZEICHNUNG
sdl.SetTextInputRect – legt das Rechteck fest, das für Unicode-Texteingaben verwendet wird

ÜBERSICHT
`sdl.SetTextInputRect(x, y, width, height)`

BESCHREIBUNG
Verwenden Sie diesen Befehl, um das Rechteck für die Eingabe von Unicode-Texteingaben festzulegen.

EINGABEN

`x`	x-Position
`y`	y-Position
`width`	Rechteckbreite
`height`	Rechteckhöhe

6.2 sdl.StartTextInput

BEZEICHNUNG
sdl.StartTextInput – beginnt Unicode-Text-Ereignisse zu empfangen

ÜBERSICHT
`sdl.StartTextInput()`

BESCHREIBUNG
Dieser Befehl akzeptiert im fokussierten RebelSDL-Fenster Unicode-Texteingabe-Ereignisse und beginnt mit der Ausgabe von `VanillaKey`. Bitte verwenden Sie diesen Befehl paarweise mit `sdl.StopTextInput()`.

Auf einigen Plattformen aktiviert dieser Befehl die Bildschirmtastatur.

EINGABEN
keine

6.3 sdl.StopTextInput

BEZEICHNUNG
sdl.StopTextInput – hört auf Unicode-Text-Ereignisse zu empfangen

ÜBERSICHT
`sdl.StopTextInput()`

BESCHREIBUNG

Verwenden Sie diesen Befehl, um keine Unicode-Texteingabe-Ereignisse mehr zu empfangen. Siehe Abschnitt 6.2 [sdl.StartTextInput], Seite 25, für Details.

EINGABEN

keine

7 Wiedergabe-Referenz

7.1 sdl.EnableOffscreenRender

BEZEICHNUNG

sdl.EnableOffscreenRender – aktiviert die Hardware-Offscreen-Wiedergabe

ÜBERSICHT

```
sdl.EnableOffscreenRender(on)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Offscreen-Wiedergabe für Hardwarepinsel zu aktivieren oder zu deaktivieren, je nachdem, was Sie im `on`-Argument übergeben haben.

Beachten Sie, dass sich die neue Einstellung nur auf Hardwarepinsel auswirkt, die nach diesem Aufruf erstellt wurden. Hardwarepinsel, die vor dem Aufruf von diesem Befehl erstellt wurden, verwenden die alten Einstellungen.

Beachten Sie auch, dass Hardwarepinsel gezeichnet werden können, um einigen Einschränkungen zu begegnen. Deshalb sollten Sie diesen Befehl nur verwenden, wenn Ihr Skript mit diesen Einschränkungen umgehen kann. Der Vorteil ist, dass `sdl.EnableOffscreenRender()` es Ihrem Skript ermöglicht, auf Hardwarepinsel mit Hardwarebeschleunigung zu zeichnen. Aber dabei müssen Sie einige Dinge im Hinterkopf behalten. Siehe Abschnitt 3.5 [Offscreen-Wiedergabe], Seite 13, für Details.

EINGABEN

`on`	boolescher Wert, der angibt, ob Hardware-Offscreen-Wiedergabe aktiviert oder deaktiviert werden soll

BEISPIEL

```
sdl.EnableOffscreenRender(True)
CreateBrush(1, 640, 480, #BLACK, {Hardware = True})
sdl.EnableOffscreenRender(False)
```

Der obige Code erstellt Pinsel 1 als einen Hardwarepinsel, der mit `SelectBrush()` gezeichnet werden kann. Alle anderen Hardwarepinsel können nicht gezeichnet werden, da wir das Flag für das Aktivieren der Offscreen-Wiedergabe sofort wieder auf `False` setzen.

7.2 sdl.GetCurrentRenderDriver

BEZEICHNUNG

sdl.GetCurrentRenderDriver – gibt den aktuellen Wiedergabe-Treiber zurück

ÜBERSICHT

```
d$ = sdl.GetCurrentRenderDriver(display)
```

BESCHREIBUNG

Dieser Befehl gibt den Namen des aktuellen Wiedergabe-Treibers zurück, z.B. `opengl`, `direct3d`, `opengles`, usw.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe-Treiber abgerufen werden soll

RÜCKGABEWERTE

`d$` Name des aktuellen Wiedergabe-Treiber

7.3 sdl.GetRenderDrawBlendMode

BEZEICHNUNG

sdl.GetRenderDrawBlendMode – gibt den Mischmodus für Zeichenoperationen zurück

ÜBERSICHT

```
blendmode = sdl.GetRenderDrawBlendMode(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Mischmodus für Zeichenoperationen zu erhalten. Siehe Abschnitt 7.24 [sdl.SetRenderDrawBlendMode], Seite 38, für eine Liste von Mischmodi.

EINGABEN

`display` Identifikator des Displays, dessen Mischmodus abgerufen werden soll

RÜCKGABEWERTE

`blendmode`
aktueller Mischmodus

7.4 sdl.GetRenderDrawColor

BEZEICHNUNG

sdl.GetRenderDrawColor – gibt die Zeichnungsfarbe zurück

ÜBERSICHT

```
r, g, b, a = sdl.GetRenderDrawColor(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Farbe für Zeichenoperationen (rect, line und clear) zu erhalten.

EINGABEN

`display` Identifikator des Displays, dessen Zeichnungsfarbe abgerufen werden soll

RÜCKGABEWERTE

`r` der rote Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`g` der grüne Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`b` der blaue Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`a` der Alpha-Wert, der verwendet wird (von 0 bis 255)

7.5 sdl.GetRendererOutputSize

BEZEICHNUNG

sdl.GetRendererOutputSize – gibt die Wiedergabe-Ausgabegröße zurück

ÜBERSICHT

```
w, h = sdl.GetRendererOutputSize(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Ausgabegröße in Pixeln eines Wiedergabekontextes zu erhalten. Wenn ein Fehler auftritt, wird in beiden Rückgabewerten -1 zurückgegeben.

EINGABEN

`display` Identifikator des Displays, dessen Ausgabegröße abgerufen werden soll

RÜCKGABEWERTE

`w` Ausgabebreite

`h` Ausgabehöhe

7.6 sdl.GetTextureAlphaMod

BEZEICHNUNG

sdl.GetTextureAlphaMod – gibt die Textur-Alpha-Modulation zurück

ÜBERSICHT

```
alpha = sdl.GetTextureAlphaMod(tex)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den zusätzlichen Alpha-Wert in Wiedergabekopiervorgänge zu multiplizieren. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

EINGABEN

`tex` Identifikator des Hardwarepinsels

RÜCKGABEWERTE

`alpha` Alpha-Modulation oder -1 im Fehlerfall

7.7 sdl.GetTextureBlendMode

BEZEICHNUNG

sdl.GetTextureBlendMode – gibt den Textur-Mischmodus zurück

ÜBERSICHT

```
mode = sdl.GetTextureBlendMode(tex)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den für Texturkopiervorgänge verwendeten Mischmodus zu erhalten. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

Siehe Abschnitt 7.24 [sdl.SetRenderDrawBlendMode], Seite 38, für eine Liste von Mischmodi.

EINGABEN

`tex` Identifikator des Hardwarepinsels

RÜCKGABEWERTE

`mode` der Mischmodus oder -1 im Fehlerfall

7.8 sdl.GetTextureColorMod

BEZEICHNUNG

sdl.GetTextureColorMod – gibt die Textur-Farbmodulation zurück

ÜBERSICHT

```
r, g, b = sdl.GetTextureColorMod(tex)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den zusätzlichen Farbwert in Wiedergabekopiervorgängen zu multiplizieren. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

Bei Fehlern gibt dieser Befehl in allen Rückgabewerten -1 zurück.

EINGABEN

`tex` Identifikator des Hardwarepinsels

RÜCKGABEWERTE

`r` der rote Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

`g` der grüne Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

`b` der blaue Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

7.9 sdl.RenderClear

BEZEICHNUNG

sdl.RenderClear – löscht das Wiedergabeziel

ÜBERSICHT

```
sdl.RenderClear(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um das aktuelle Wiedergabeziel mit der Zeichenfarbe zu löschen. Dieser Befehl löscht das gesamte Wiedergabeziel und ignoriert das Ansichtsfenster und das Cliprechteck.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

7.10 sdl.RenderCopy

BEZEICHNUNG

sdl.RenderCopy – zeichnet eine Textur

ÜBERSICHT

```
sdl.RenderCopy(display, tex[, src, dst, angle, center, flip])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um einen Teil der von `tex` angegebenen Textur in das von `display` angegebene Display zu kopieren, wobei Sie sie optional um den angegebenen Mittelpunkt `center` drehen und auch von oben nach unten und/oder von links nach rechts spiegeln können. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

Wenn angegeben, müssen `src` und `dst` Tabellen sein, die die folgenden Felder enthalten:

`x` Linke Position des Rechtecks.

`y` Oberste Position des Rechtecks.

`w` Rechteckbreite.

`h` Rechteckhöhe.

Alternativ können Sie auch `src` und/oder `dst` auf `Nil` setzen, somit wird die gesamte Größe der Quelle bzw. des Ziels verwendet. Wenn Quell- und Zielgröße nicht übereinstimmen, dehnt `sdl.RenderCopy()` die Textur automatisch auf das Zielrechteck aus.

Wenn angegeben, muss `center` eine Tabelle sein, die die folgenden Felder enthält:

`x` Linke Position des Mittelpunkts.

`y` Oberste Position des Mittelpunkts.

Alternativ können Sie auch `center` auf Null setzen. In diesem Fall wird der Mittelpunkt auf das Zentrum des Zielrechtecks gesetzt.

Wenn angegeben, muss `flip` eine Kombination der folgenden vordefinierten Konstanten sein:

`#SDL_FLIP_NONE`
: Nicht spiegeln

`#SDL_FLIP_HORIZONTAL`
: Horizontal spiegeln

`#SDL_FLIP_VERTICAL`
: Vertikal spiegeln

Die Textur wird mit dem Ziel gemischt, basierend auf dem Mischmodus, der mit `sdl.SetTextureBlendMode()` eingestellt wurde.

Die Texturfarbe wird basierend auf ihrer Farbmodulation beeinflusst, die durch `sdl.SetTextureColorMod()` eingestellt wurde.

Textur-Alpha wird basierend auf seiner Alphamodulation beeinflusst, die durch `sdl.SetTextureAlphaMod()` eingestellt wurde.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`tex`	Identifikator des Hardwarepinsels
`src`	optional: Rechteck in der zu verwendenden Textur (standardmäßig `Nil`, d.h. die gesamte Textur wird verwendet)
`dst`	optional: Rechteck zum Zeichnen ins Display (standardmäßig `Nil`, d.h. füllt das gesamte Display aus)
`angle`	optional: Winkel in Grad, der im Uhrzeigersinn auf das Zielrechteck angewendet wird (standardmäßig 0)
`center`	optional: Punkt um den das Zielrechteck gedreht werden soll (Standardwert ist Null, d.h. Zielrechteckmitte)
`flip`	optional: Dreh-Aktionen, die ausgeführt werden sollen (standardmäßig auf `#SDL_FLIP_NONE`)

7.11 sdl.RenderDrawLine

BEZEICHNUNG

sdl.RenderDrawLine – zeichnet eine Linie

ÜBERSICHT

```
sdl.RenderDrawLine(display, x1, y1, x2, y2)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um eine Linie auf dem aktuellen Wiedergabeziel zu zeichnen.

Die aktuelle Zeichnungsfarbe wird durch `sdl.SetRenderDrawColor()` festgelegt und der Alpha-Wert der Farbe wird ignoriert, sofern die Mischung nicht mit dem entsprechenden Aufruf von `sdl.SetRenderDrawBlendMode()` aktiviert wurde.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`x1`	die x-Koordinate des Startpunkts
`y1`	die y-Koordinate des Startpunkts
`x2`	die x-Koordinate des Endpunkts
`y2`	die y-Koordinate des Endpunkts

7.12 sdl.RenderDrawPoint

BEZEICHNUNG

sdl.RenderDrawPoint – zeichnet einen Punkt

ÜBERSICHT

```
sdl.RenderDrawPoint(display, x, y)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um einen Punkt auf dem aktuellen Wiedergabeziel zu zeichnen.

Die aktuelle Zeichnungsfarbe wird durch `sdl.SetRenderDrawColor()` festgelegt und der Alpha-Wert der Farbe wird ignoriert, sofern die Mischung nicht mit dem entsprechenden Aufruf von `sdl.SetRenderDrawBlendMode()` aktiviert wurde.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`x`	die x-Koordinate des Punktes
`y`	die y-Koordinate des Punktes

7.13 sdl.RenderDrawRect

BEZEICHNUNG

sdl.RenderDrawRect – zeichnet ein Rechteck

ÜBERSICHT

```
sdl.RenderDrawRect(display[, x, y, w, h])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um ein Rechteck auf dem aktuellen Wiedergabeziel zu zeichnen. Wenn Sie die optionalen Argumente weglassen, wird das gesamte Wiedergabeziel umrandet.

Die aktuelle Zeichnungsfarbe wird durch `sdl.SetRenderDrawColor()` festgelegt und der Alpha-Wert der Farbe wird ignoriert, sofern die Mischung nicht mit dem entsprechenden Aufruf von `sdl.SetRenderDrawBlendMode()` aktiviert wurde.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`x`	optional: Die x-Koordinate der oberen linken Ecke
`y`	optional: Die y-Koordinate der oberen linken Ecke
`w`	optional: Die Rechteckbreite
`h`	optional: Die Rechteckhöhe

7.14 sdl.RenderFillRect

BEZEICHNUNG

sdl.RenderFillRect – zeichnet ein gefülltes Rechteck

ÜBERSICHT

```
sdl.RenderFillRect(display[, x, y, w, h])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um ein Rechteck auf dem aktuellen Wiedergabeziel zu zeichnen. Wenn Sie die optionalen Argumente weglassen, wird das gesamte Wiedergabeziel gefüllt.

Die aktuelle Zeichnungsfarbe wird durch `sdl.SetRenderDrawColor()` festgelegt und der Alpha-Wert der Farbe wird ignoriert, sofern die Mischung nicht mit dem entsprechenden Aufruf von `sdl.SetRenderDrawBlendMode()` aktiviert wurde.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`x`	optional: Die x-Koordinate der oberen linken Ecke
`y`	optional: Die y-Koordinate der oberen linken Ecke
`w`	optional: Die Rechteckbreite
`h`	optional: Die Rechteckhöhe

7.15 sdl.RenderGetClipRect

BEZEICHNUNG

sdl.RenderGetClipRect – gibt den rechteckigen Ausschnitt eines Displays zurück

ÜBERSICHT

```
x, y, w, h = sdl.RenderGetClipRect(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Rechteck-Ausschnitt für das aktuelle Ziel zu erhalten.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll

RÜCKGABEWERTE

`x`	die x-Koordinate der oberen linken Ecke
`y`	die y-Koordinate der oberen linken Ecke
`w`	die Rechteckbreite
`h`	die Rechteckhöhe

7.16 sdl.RenderGetLogicalSize

BEZEICHNUNG

sdl.RenderGetLogicalSize – ermittelt die logische Größe

ÜBERSICHT

```
w, h = sdl.RenderGetLogicalSize(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die geräteunabhängige Auflösung für die Wiedergabe zu erhalten. Wenn dieser Befehl für die Wiedergabe aufgerufen wird, dessen logische Größe nicht durch `sdl.RenderSetLogicalSize()` festgelegt wurde, gibt dieser Befehl sowohl in `w` als auch in `h` 0 zurück.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

RÜCKGABEWERTE

`w` die Breite der logischen Auflösung

`h` die Höhe der logischen Auflösung

7.17 sdl.RenderGetScale

BEZEICHNUNG

sdl.RenderGetScale – ermittelt den Skalierungsfaktoren

ÜBERSICHT

```
scalex, scaley = sdl.RenderGetScale(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Skalierungsfaktor für das aktuelle Ziel zu erhalten.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

RÜCKGABEWERTE

`scalex` der horizontale Skalierungsfaktor

`scaley` der vertikale Skalierungsfaktor

7.18 sdl.RenderGetViewport

BEZEICHNUNG

sdl.RenderGetViewport – ermittelt den Zeichnungsbereich

ÜBERSICHT

```
x, y, w, h = sdl.RenderGetViewport(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Zeichnungsbereich für das aktuelle Ziel zu erhalten.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

RÜCKGABEWERTE

`x` die x-Koordinate der oberen linken Ecke

`y` die y-Koordinate der oberen linken Ecke

`w` die Rechteckbreite

`h` die Rechteckhöhe

7.19 sdl.RenderPresent

BEZEICHNUNG

sdl.RenderPresent – zeigt das Einzelbild aus dem Hintergrundpuffer an

ÜBERSICHT

```
sdl.RenderPresent(display)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Bildschirm mit jeder Wiedergabe seit dem letzten Aufruf zu aktualisieren.

SDLs Wiedergabe-Befehle arbeiten mit einem Hintergrundpuffer; das heißt, ein Wiedergabe-Befehl wie `sdl.RenderDrawLine()` ruft nicht direkt eine Zeile auf dem Bildschirm auf, sondern aktualisiert den Puffer im Hintergrund. Als solches komponieren Sie Ihre gesamte Szene und präsentieren den komponierten Hintergrundpuffer auf dem Bildschirm als vollständiges Bild.

Wenn Sie also die Wiedergabe-API von SDL verwenden, werden alle für das Einzelbild vorgesehene Zeichnungen ausgeführt. Anschließend wird dieser Befehl einmal pro Einzelbild aufgerufen, um dem Benutzer die endgültige Zeichnung anzuzeigen.

Der Hintergrundpuffer sollte nach jeder Präsentation für ungültig erklärt werden. Gehen Sie nicht davon aus, dass der vorherige Inhalt zwischen den Einzelbildern bestehen wird. Es wird dringend empfohlen, `sdl.RenderClear()` aufzurufen, um den Hintergrundpuffer vor Beginn der Zeichnung jedes neuen Einzelbildes zu initialisieren, selbst wenn Sie jedes Pixel überschreiben möchten.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

7.20 sdl.RenderSetClipRect

BEZEICHNUNG

sdl.RenderSetClipRect – setzt den rechteckigen Ausschnitt

ÜBERSICHT

```
sdl.RenderSetClipRect(display[, x, y, w, h])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den rechteckigen Ausschnitt für die Wiedergabe auf dem angegebenen Ziel festzulegen. Wenn Sie die optionalen Argumente weglassen, wird der Ausschnitt deaktiviert.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

`x`	die x-Koordinate der oberen linken Ecke
`y`	die y-Koordinate der oberen linken Ecke
`w`	die Rechteckbreite
`h`	die Rechteckhöhe

7.21 sdl.RenderSetLogicalSize

BEZEICHNUNG

sdl.RenderSetLogicalSize – stellt die logische Größe ein

ÜBERSICHT

```
sdl.RenderSetLogicalSize(display, w, h)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um eine geräteunabhängige Auflösung für die Wiedergabe festzulegen.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`w`	die Breite der logischen Auflösung
`h`	die Höhe der logischen Auflösung

7.22 sdl.RenderSetScale

BEZEICHNUNG

sdl.RenderSetScale – stellt den Skalierungsfaktor ein

ÜBERSICHT

```
sdl.RenderSetScale(display, scalex, scaley)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Skalierungsfaktor für die Wiedergabe auf dem aktuellen Ziel festzulegen. Die Zeichenkoordinaten werden mit den x/y-Skalierungsfaktoren skaliert, bevor sie bei der Wiedergabe verwendet werden. Dies ermöglicht eine auflösungsunabhängige Zeichnung mit einem einzigen Koordinatensystem.

Wenn dies zu einer Skalierung oder Subpixelzeichnung durch das Rendering-Backend führt, wird dies mit den entsprechenden Qualitätshinweisen gehandhabt. Für beste Ergebnisse verwenden Sie Integer-Skalierungsfaktoren.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`scalex`	der horizontale Skalierungsfaktor
`scaley`	der vertikale Skalierungsfaktor

7.23 sdl.RenderSetViewport

BEZEICHNUNG

sdl.RenderSetViewport – legt den Zeichnungsbereich fest

ÜBERSICHT

```
sdl.RenderSetViewport(display[, x, y, w, h])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Zeichnungsbereich für die Wiedergabe auf dem aktuellen Ziel festzulegen. Wenn Sie die optionalen Argumente weglassen, wird der Zeichnungsbereich auf das gesamte Ziel festgelegt. Wenn das Fenster in der Größe geändert wird, wird der aktuelle Zeichnungsbereich automatisch in der neuen Fenstergröße zentriert.

EINGABEN

`display`	Identifikator des Displays, dessen Wiedergabe verwendet werden soll
`x`	optional: Die x-Koordinate der oberen linken Ecke
`y`	optional: Die y-Koordinate der oberen linken Ecke
`w`	optional: Die Rechteckbreite
`h`	optional: Die Rechteckhöhe

7.24 sdl.SetRenderDrawBlendMode

BEZEICHNUNG

sdl.SetRenderDrawBlendMode – stellt den Mischmodus ein

ÜBERSICHT

```
sdl.SetRenderDrawBlendMode(display, blendmode)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Mischmodus für Zeichnungsoperationen (Füllung und Linie) festzulegen.

`#SDL_BLENDMODE_NONE`
: Keine Mischung

```
dstRGBA = srcRGBA
```

`#SDL_BLENDMODE_BLEND`
: Alpha Mischung

```
dstRGB = (srcRGB * srcA) + (dstRGB * (1-srcA))
dstA = srcA + (dstA * (1-srcA))
```

`#SDL_BLENDMODE_ADD`
: Zusatz Mischung

```
dstRGB = (srcRGB * srcA) + dstRGB
dstA = dstA
```

`#SDL_BLENDMODE_MOD`
: Farbe modulieren

```
dstRGB = srcRGB * dstRGB
dstA = dstA
```

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

`blendmode`
: einer der Mischmodi von oben

7.25 sdl.SetRenderDrawColor

BEZEICHNUNG

sdl.SetRenderDrawColor – stellt die Zeichnungsfarbe ein

ÜBERSICHT

```
sdl.SetRenderDrawColor(display, r, g, b[, a])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um die Farbe zum Zeichnen oder Ausfüllen von Rechtecken, Linien und Punkten sowie für `sdl.RenderClear()` einzustellen.

Verwenden Sie `sdl.SetRenderDrawBlendMode()`, wenn Sie Alpha-Mischung benutzen möchten, um anzugeben, wie der Alphakanal verwendet wird.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

`r` der rote Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`g` der grüne Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`b` der blaue Wert, der zum Zeichnen verwendet wird (von 0 bis 255)

`a` optional: der Alpha-Wert, der verwendet wird, um auf das Wiedergabe-Ziel zu zeichnen (standardmäßig `#SDL_ALPHA_OPAQUE`)

7.26 sdl.SetRenderTarget

BEZEICHNUNG

sdl.SetRenderTarget – legt das Wiedergabeziel fest

ÜBERSICHT

```
sdl.SetRenderTarget(display[, tex])
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um eine Textur als aktuelles Wiedergabeziel festzulegen. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein. Wenn Sie das Argument `tex` weglassen, wird das Standard-Wiedergabeziel verwendet.

EINGABEN

`display` Identifikator des Displays, dessen Wiedergabe verwendet werden soll

`tex` optional: Hardwarepinsel, der als Ziel verwendet wird

7.27 sdl.SetScaleQuality

BEZEICHNUNG

sdl.SetScaleQuality – stellt die Qualität der Texturskalierung ein (V1.1)

ÜBERSICHT

```
sdl.SetScaleQuality(q$)
```

BESCHREIBUNG

Mit diesem Befehl können Sie festlegen, wie Texturen (oder Hollywood-Hardwarepinsel, die intern als Texturen gespeichert werden) skaliert werden sollen. Sie müssen eine Zeichenkette im Argument `q$` übergeben. Die folgenden Skalierungsmodi werden derzeit unterstützt:

`nearest:` Ohne Interpolation.

`linear:` Lineare Filterung.

`best:` Anisotrope Filterung. Dies wird derzeit nur unter Windows mit Direct 3D unterstützt.

Beachten Sie, dass der Skalierungsmodus festgelegt wird, wenn eine Textur (oder ein Hardware-Pinsel) erstellt wird. Sie müssen daher `sdl.SetScaleQuality()` aufrufen, bevor Sie einen Hardware-Pinsel / eine Hardware-Textur erstellen. Dies bedeutet auch, dass Sie es nicht für Pinsel verwenden können, die mit den Präprozessor-Anweisungen geladen werden, da diese immer vor Beginn der Skriptausführung geladen werden.

EINGABEN

`q$` gewünschte Skalierungsqualität (mögliche Werte siehe oben)

7.28 sdl.SetTextureAlphaMod

BEZEICHNUNG

sdl.SetTextureAlphaMod – setzt die Textur-Alpha-Modulation

ÜBERSICHT

```
r = sdl.SetTextureAlphaMod(tex, alpha)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um einen zusätzlichen Alpha-Wert multipliziert mit Wiedergabekopiervorgängen einzustellen. Wenn diese Textur ausgegeben wird, wird der Alpha-Quellwert während des Kopiervorgangs durch diesen Alpha-Wert gemäß der folgenden Formel moduliert:

```
srcA = srcA * (alpha / 255)
```

Beachten Sie, dass eine Alpha-Modulation nicht immer von der Wiedergabe unterstützt wird. Es wird -1 zurückgegeben, wenn die Alpha-Modulation nicht unterstützt wird.

Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

EINGABEN

`tex` Identifikator des Hardwarepinsels

`alpha` der Quell-Alpha-Wert multipliziert mit Kopiervorgängen (von 0 bis 255)

RÜCKGABEWERTE

`r` 0 bei Erfolg oder ein negativer Fehlercode bei Fehler

7.29 sdl.SetTextureBlendMode

BEZEICHNUNG

sdl.SetTextureBlendMode – stellt den Textur-Mischmodus ein

ÜBERSICHT

```
r = sdl.SetTextureBlendMode(tex, blendmode)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Mischmodus für eine Textur festzulegen, die von `sdl.RenderCopy()` verwendet wird.

`blendmode` kann eine der folgenden Konstanten sein:

`#SDL_BLENDMODE_NONE`
Keine Mischung

```
dstRGBA = srcRGBA
```

`#SDL_BLENDMODE_BLEND`
Alpha-Mischung

```
dstRGB = (srcRGB * srcA) + (dstRGB * (1-srcA))
dstA = srcA + (dstA * (1-srcA))
```

`#SDL_BLENDMODE_ADD`
Zusatz Mischung

```
dstRGB = (srcRGB * srcA) + dstRGB
dstA = dstA
```

`#SDL_BLENDMODE_MOD`
Farbe modulieren

```
dstRGB = srcRGB * dstRGB
dstA = dstA
```

Wenn der Mischmodus nicht unterstützt wird, wird der nächstliegende unterstützte Modus ausgewählt und dieser Befehl gibt -1 zurück. Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

EINGABEN

`tex` Identifikator des Hardwarepinsels

`blendmode`
: der Mischmodus für die Textur-Mischung (siehe oben)

RÜCKGABEWERTE

`r` 0 bei Erfolg oder ein negativer Fehlercode bei Fehler

7.30 sdl.SetTextureColorMod

BEZEICHNUNG

sdl.SetTextureColorMod – stellt die Textur-Farbmodulation ein

ÜBERSICHT

```
r = sdl.SetTextureColorMod(tex, r, g, b)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um einen zusätzlichen Farbwert multipliziert mit Wiedergabekopiervorgängen einzustellen. Wenn diese Textur ausgegeben wird, wird während des Kopiervorgangs jeder Quellfarbkanal durch den entsprechenden Farbwert gemäß der folgenden Formel moduliert:

```
srcC = srcC * (color / 255)
```

Farbmodulation wird von der Wiedergabe nicht immer unterstützt. Es wird -1 zurückgegeben, wenn die Farbmodulation nicht unterstützt wird.

Das Argument `tex` muss einfach der Identifikator eines Hardwarepinsels sein.

EINGABEN

`tex` Identifikator des Hardwarepinsels

`r` der rote Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

`g` der grüne Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

`b` der blaue Farbwert multipliziert mit Kopiervorgängen (von 0 - 255)

RÜCKGABEWERTE

`r` 0 bei Erfolg oder ein negativer Fehlercode bei Fehler

8 System-Referenz

8.1 sdl.ClearError

BEZEICHNUNG
sdl.ClearError – löscht den letzten Fehler

ÜBERSICHT
`sdl.ClearError()`

BESCHREIBUNG
Verwenden Sie diesen Befehl, um vorherige Fehlermeldungen zu löschen.

EINGABEN
keine

8.2 sdl.GetCurrentVideoDriver

BEZEICHNUNG
sdl.GetCurrentVideoDriver – gibt den aktuellen Videotreiber zurück

ÜBERSICHT
`d$ = sdl.GetCurrentVideoDriver()`

BESCHREIBUNG
Dieser Befehl gibt den Namen des aktuellen Videotreibers zurück, z.B. `windows`, `cocoa`, `x11`, usw.

EINGABEN
keine

RÜCKGABEWERTE
`d$` Name des aktuellen Videotreibers

8.3 sdl.GetError

BEZEICHNUNG
sdl.GetError – gibt den letzten Fehler zurück

ÜBERSICHT
`e$ = sdl.GetError()`

BESCHREIBUNG
Gibt eine Nachricht mit Informationen über den aufgetretenen Fehler oder eine leere Zeichenkette zurück, wenn seit dem letzten Aufruf von `sdl.ClearError()` keine Fehlernachricht festgelegt wurde. Die Nachricht ist nur anwendbar, wenn ein SDL-Befehl einen Fehler gemeldet hat. Sie müssen die Rückgabewerte von SDL-Befehlsaufrufen überprüfen, um zu bestimmen, wann `sdl.GetError()` entsprechend aufgerufen werden soll.

EINGABEN
keine

RÜCKGABEWERTE

`e$`	letzte Fehlermeldung oder leere Zeichenkette

8.4 sdl.GetVersion

BEZEICHNUNG
sdl.GetVersion – ermittelt die SDL-Version

ÜBERSICHT
```
ver, rev, patch = sdl.GetVersion()
```

BESCHREIBUNG
Verwenden Sie diesen Befehl, um die Version von SDL abzurufen, die mit Ihrem Programm verknüpft ist.

EINGABEN
keine

RÜCKGABEWERTE

`ver`	Hauptversion
`rev`	Nebenversion
`patch`	Update-Version (Patch-Level)

9 Fenster-Referenz

9.1 sdl.SetWindowFullscreen

BEZEICHNUNG

sdl.SetWindowFullscreen – wechselt den Anzeigemodus

ÜBERSICHT

```
sdl.SetWindowFullscreen(display, mode)
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um den Vollbildstatus eines Fensters festzulegen.

`mode` muss eine der folgenden vordefinierten Konstanten sein:

`#SDL_WINDOW_FULLSCREEN`
: Echtes Vollbild mit einem Videomoduswechsel

`#SDL_WINDOW_FULLSCREEN_DESKTOP`
: Gefälschtes Vollbild, der die Größe des Desktops annimmt

`#SDL_WINDOW_WINDOW`
: Fenstermodus

EINGABEN

`display`	Identifikator des Displays, dessen Anzeigemodus gewechselt werden soll
`mode`	neuer Anzeigemodus

Anhang A Lizenzen

A.1 SDL-Lizenzen

Index

sdl.ClearError 43
sdl.EnableOffscreenRender 27
sdl.ForceJoystickMode 19
sdl.GetAxis 19
sdl.GetBall 20
sdl.GetButton 20
sdl.GetCurrentRenderDriver 27
sdl.GetCurrentVideoDriver 43
sdl.GetError 43
sdl.GetHat 21
sdl.GetJoysticks 22
sdl.GetNumAxes 22
sdl.GetNumBalls 22
sdl.GetNumButtons 23
sdl.GetNumHats 23
sdl.GetRenderDrawBlendMode 28
sdl.GetRenderDrawColor 28
sdl.GetRendererOutputSize 28
sdl.GetTextureAlphaMod 29
sdl.GetTextureBlendMode 29
sdl.GetTextureColorMod 30
sdl.GetVersion 44
sdl.IsGameController 23
sdl.RenderClear 30
sdl.RenderCopy 30
sdl.RenderDrawLine 32
sdl.RenderDrawPoint 32
sdl.RenderDrawRect 33
sdl.RenderFillRect 33
sdl.RenderGetClipRect 34
sdl.RenderGetLogicalSize 34
sdl.RenderGetScale 35
sdl.RenderGetViewport 35
sdl.RenderPresent 36
sdl.RenderSetClipRect 36
sdl.RenderSetLogicalSize 37
sdl.RenderSetScale 37
sdl.RenderSetViewport 37
sdl.SetRenderDrawBlendMode 38
sdl.SetRenderDrawColor 39
sdl.SetRenderTarget 39
sdl.SetScaleQuality 40
sdl.SetTextInputRect 25
sdl.SetTextureAlphaMod 40
sdl.SetTextureBlendMode 41
sdl.SetTextureColorMod 42
sdl.SetWindowFullscreen 45
sdl.StartTextInput 25
sdl.StopTextInput 25